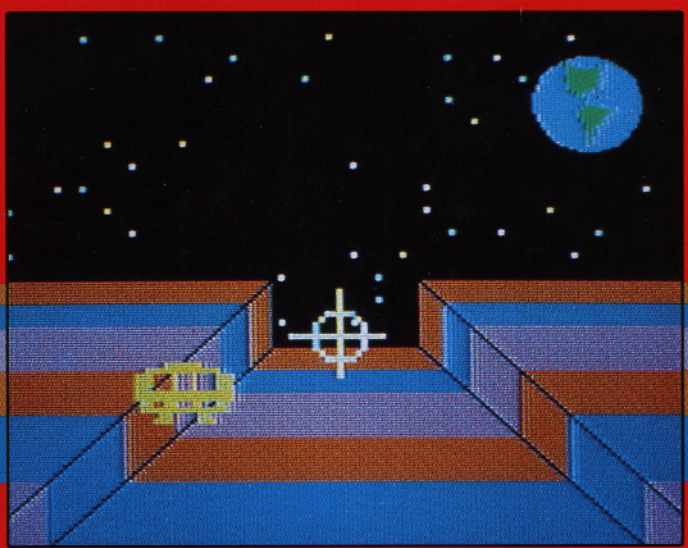
COMPUTE!'s FIRST BOOK OF

TI GAMES

29 action and learning games,
including the best from *COMPUTE!* Magazine
plus many never before published.



Edited with an introduction and notes by C. Regena

A **COMPUTE! Books** Publication

$12.95

COMPUTE!'s FIRST BOOK OF

TI GAMES

Edited with an introduction and notes by C. Regena

COMPUTE! Publications, Inc.
One of the ABC Publishing Companies
Greensboro, North Carolina

The following article was originally published in *COMPUTE!* Magazine, copyright 1981, Small System Services, Inc.:
"Maze Generator" (December)

The following articles were originally published in *COMPUTE!* Magazine, copyright 1982, Small System Services, Inc.:
"MathMan" (October)
"Superchase" (October)
"Hidden Maze" (December)

The following articles were originally published in *COMPUTE!* Magazine, copyright 1983, Small System Services, Inc.:
"Copy Cat" (February)
"Trapshoot" (March)
"Closeout" (March)
"Boggler" (March)
"Match-Em" (April)
"Air Defense" (April)
"Jumping Jack" (May)

The following articles were originally published in *COMPUTE!* Magazine, copyright 1983, COMPUTE! Publications, Inc.:
"Astrostorm" (June)
"Memory Trainer" (June)
"Goblin" (July)
"Diamond Drop" (September)
"Mystery Spell" (September)
"Devastator" (October)
"Mosaic Puzzle" (October)

The following article was originally published in *COMPUTE!'s First Book of VIC Games*, copyright 1983, COMPUTE! Publications, Inc.:
"Marble Hunt"

Printed in the United States of America

ISBN 0-942386-17-5

10 9 8 7 6 5 4 3 2 1

COMPUTE! Publications, Inc., Post Office Box 5406, Greensboro, NC 27403, (919) 275-9809, is a subsidiary of American Broadcasting Companies, Inc., and is not associated with any manufacturer of personal computers. TI-99/4 and TI-99/4A are trademarks of Texas Instruments.

Contents

Foreword

One of the nicest things about the TI-99/4A is that it isn't just a useful home computer—it can also be a lot of fun. Here are 30 games that will help you turn your TI-99/4A into a home arcade and learning center. All you have to do is type them in and play.

The editor, C. Regena, is the author of *Programmer's Reference Guide to the TI-99/4A* and *COMPUTE!* Magazine's regular columnist for the TI-99/4A. If you know her from those publications, you're already familiar with her clear, thorough explanations and resourceful programming.

In *COMPUTE!'s First Book of TI Games*, she has collected the best games that have appeared in *COMPUTE!*—plus many more that she either wrote or adapted from programs for other computers. Her clear explanations make them easy to play, and the introductory chapters will help you understand the principles of game programming so you can adapt these programs or create your own computer games.

Whether you plan to learn more about programming or just want to add games to your software library, you will find the quality of this book and its programs consistent with the quality of all COMPUTE! publications.

Our thanks to the many members of our editorial and production staff who assisted in the development of this book.

Introduction

Welcome to the entertaining world of games on your TI-99/4 or TI-99/4A home computer. A computer is ideal for games for two main reasons. First, a computer can repeat a procedure many times without getting tired, and a computer can sift information very quickly to decide the next move in a game. Second, a computer has a random number generator so the game can be different every time.

If you like to play games, try typing in some from this book, which contains a variety of games written specifically for the TI computer. The only peripheral needed is a cassette recorder to store the programs. Most of the games included here require no extra purchases of equipment. An exception is the TI Extended BASIC command module. Extended BASIC allows the use of smoothly moving sprites; therefore, a few Extended BASIC games are included in Chapter 9 of the book.

For each game you will find a description of the game with instructions and the actual listing of the program. Most games also include an explanation of the program with techniques used.

If you enjoy programming, you will be able to learn some programming techniques from these games. Study these techniques and experiment. Change the graphics to a different theme, make a simple game more complex, use different sound effects, put more options in the game, add constraints, improve the logic in computer versus player games—create your own "best seller."

1

Programming the TI

1

Special Features of the TI

Graphics. The full screen for the TI computer is 24 rows by 32 columns. Each square in the 24 x 32 rectangle can be divided up into an 8 x 8 square for high resolution. If you have a lot of special graphic characters, you may redefine any of the standard characters.

Color. You may use all 16 available colors on the screen at the same time. You can get a different screen color by using the CALL SCREEN statement. Each high-resolution square can be any two colors. Color sets can be planned to create flashing or blinking. Objects may be drawn invisibly then flashed on all at once.

Sounds. You may play one, two, or three tones at a time to develop music to fit your game theme. Noise sounds can also be created, and you can combine noise sounds with up to three tones for all sorts of effects. Use sounds in FOR-NEXT loops for even more special effects.

Arrow keys. The arrow keys are printed on the face of the keys for E, S, D, and X, so you can get used to standard arrow keys for all games. The most common keys for firing are the ENTER key, the period, and the space bar.

Joysticks. Dual joysticks are available. These have two joysticks with one plug, so you can play one- or two-player games.

Detection. To determine what character is in a certain screen location, the CALL GCHAR command is used. This feature is useful in maze or obstacle games or in detecting collisions between screen objects.

Game Modules. Easy-to-use command modules are available for a wide variety of games. The modules use full capabilities of graphics and sound to create fun games. Games in modules may be programmed in Graphics Programming Language (GPL) or in machine language for the speed of arcade-style games.

Randomness. RND and RANDOMIZE are commands for creating random numbers, which make the game different every time you play.

Text/Graphics. You may print text anywhere on the screen at the same time you have high-resolution graphics.

1

Hints for Game Programming

Start with an idea. What kind of game are you going to make? Standard parlor or board games make good computer versus player games. One-player games could include mazes and obstacle courses, logic or puzzle-solving games, or a variety of space and shooting games. You might want to choose a memory game, an adventure, or perhaps a popular sport. You may prefer to pattern your game after an arcade game. There are limitless possibilities, and you can make a game really your own.

Consider the features of the computer. Use the computer to draw fun graphics and play music or create sound effects, but keep in mind that some games could just as well be played on paper or on a standard board. Use the computer for its special features. Use the computer to think out logic processes in two-player games so that it becomes one of the players. Use the randomness to roll dice, deal cards, or make a maze.

If you want fast action like that in an arcade-style game, you may need the TI Extended BASIC module, which adds the capability of smoothly moving sprites (see Chapter 9). Action games in regular console BASIC are more difficult because moving objects must be drawn, then erased, then drawn, then erased to create motion. TI BASIC lends itself a little better to the "thinking" games.

Plan the game segments. Even if you are the type of programmer who just sits at the console and starts typing in statements without first drawing plans, you should organize your thoughts and separate the game into *modules* that can later be combined. I often number all statements by tens so that I can later insert forgotten lines. I may start the title screen and character definitions with line 1000, the first player's movement with line 2000, the main game loop with line 3000, the second player's movement with line 4000, and ending procedures with line 6000. Later I can renumber the lines so that the program looks nice in a listing.

Include instructions within the program. Unless you are

limited by memory or unless the game is an old standard with obvious rules, put some brief instructions within the program. Do try to keep the instructions brief. Use a "PRESS ANY KEY" or "PRESS ENTER" method of continuing after instruction screens. Then the players can read as long as they wish—or as briefly as possible. You may wish to let the player choose whether he wants instructions or not:

```
200 PRINT "NEED INSTRUCTIONS? Y/N"
210 CALL KEY(0,K,S)
220 IF K=78 THEN 1000
230 IF K<>89 THEN 210
240 PRINT INSTRUCTIONS HERE...
1000 GAME STARTS HERE...
```

You may prefer to remind all players of the instructions with one screen of instructions, then:

```
300 PRINT "PRESS A KEY TO BEGIN.";
310 CALL KEY(0,K,S)
320 IF S<1 THEN 310
330 PROGRAM CONTINUES HERE...
```

Plan efficient motion. If you choose to create an action game, the motion has to seem as fast as possible. Keep moving objects the size of one character so you do not have to erase and draw several characters for each movement. Minimize the number of moving characters. The more characters you have moving, the longer the time between the motions of each character, and the slower the game. Plan the drawing of graphics in the most efficient way—printing is faster than using several CALL HCHAR or CALL VCHAR statements. Use the repetition factors in the CALL HCHAR or CALL VCHAR statements when possible for improved drawing speed. You may wish to draw things invisibly and then use a CALL COLOR to make the object appear all at once.

Minimize the memory requirements. In general, the more memory the game requires, the slower the game will be. Delete all unnecessary statements. Perhaps you can use REMark statements while you're developing the program, then delete them when the game is working properly. Look through your listing for repetitious logic that can be put in subroutines. Watch the use of GOTO statements. Good planning can decrease the number of transfer statements.

Test boundary conditions. If you are moving objects around the screen, be sure to test for the top, bottom, and sides of the screen so the program does not crash with a bad value in a graphics statement. Any time you are using variables, consider what will happen as the variables get larger, or if they are negative—or zero. If you have a game in which you travel to either the left or the right to increment scoring, be sure to test the scoring for both directions. You may need to use the ABS (absolute value) function if you are working with positive and negative numbers.

Check player's input. There is much less chance for error if the player can just press a single key, rather than type something and then press ENTER. If you must use the INPUT statement, the player's response needs to be tested. Keep in mind that the player's typing will scroll: will graphics be messed up if you intend the player to enter three letters but instead she or he enters four lines? If you are expecting a number, what happens if a letter is entered? If you are using numbers, you need to check on limits—large positive numbers, negative numbers, and zero.

CALL KEY is a way to detect if a single key is being pressed. You can accept only certain keys and ignore all other keys pressed. For example, if you want the player to use the arrow keys, you can check for the keys E, S, D, and X, and ignore all others—so if the player accidentally hits C, the program will not crash. Here are two examples using the CALL KEY statement:

```
100 PRINT "PRESS 1, 2, 3, 4"  (four different options)
110 CALL KEY(0,K,S)
120 IF K<49 THEN 110          (If the ASCII value of the key
130 IF K>52 THEN 110           pressed is less than the code for
140 LV=K-48                    1, ignore it; if the value is greater
     :                         than the code for 4, ignore it.)
     :
     :

900 PRINT "TRY AGAIN? (Y/N)"
910 CALL KEY(0,KEY,ST)
920 IF KEY=89 THEN 320        (If key pressed is Y, go to 320. If
930 IF KEY<>78 THEN 910        key pressed is N, end. All other
940 END                        keys are ignored.)
```

1

Specific Programming Techniques

Printing a Message on the Screen Without Scrolling

You may use the following technique to print a score SC (which is a number) in row 22 of the screen and starting in column 10:

```
100 SC$=STR$(SC)
110 FOR C=1 TO LEN(SC$)
120 CALL HCHAR(22,9+C,ASC(SEG$(SC$,C,1)))
130 NEXT C
```

In general, if you have a message M$ you want printed in row X and column Y, try the following subroutine:

```
600 FOR C=1 TO LEN(M$)
610 CALL HCHAR(X,Y-1+C,ASC(SEG$(M$,C,1)))
620 NEXT C
630 RETURN
```

To use the subroutine, here is an example:

```
1000 M$="SUPER JOB!"
1010 X=12
1020 Y=3
1030 GOSUB 600
```

Using the Arrow Keys to Move

There are many ways to program movement using the arrow keys. The program below gives one example:

```
100 REM ARROW KEYS--1
110 CALL CLEAR
120 CALL COLOR(9,7,7)
130 I=12
140 J=15
150 CALL HCHAR(I,J,96)
```

```
160 CALL KEY(0,K,S)
170 IF K<>69 THEN 220
180 I=I-1
190 IF I>0 THEN 150
200 I=1
210 GOTO 150
220 IF K<>68 THEN 270
230 J=J+1
240 IF J<33 THEN 150
250 J=32
260 GOTO 150
270 IF K<>83 THEN 320
280 J=J-1
290 IF J>0 THEN 150
300 J=1
310 GOTO 150
320 IF K<>88 THEN 160
330 I=I+1
340 IF I<25 THEN 150
350 I=24
360 GOTO 150
370 END
```

The program starts by putting a red square in the center of the screen. The CALL KEY statement in line 160 detects which key is pressed. Depending on which arrow key (E, S, D, X) is pressed, the square will move about the screen. The boundary conditions are checked to insure that the red square will stay at the edges rather than go off the screen.

Here is another example using the arrow keys. This time the object is moved by erasing the first position before drawing the new position. At the edges of the screen the object will "wrap" to the other edge. For example, if you press the up arrow and the object is at the top of the screen, it will reappear at the bottom of the screen then continue upward.

```
100 REM ARROW KEYS--2
110 CALL CLEAR
120 I=12
130 J=15
140 CALL HCHAR(I,J,64)
150 CALL KEY(0,K,S)
```

```
160 IF K<>69 THEN 200
170 DI=-1
180 DJ=0
190 GOTO 310
200 IF K<>68 THEN 240
210 DI=0
220 DJ=1
230 GOTO 310
240 IF K<>83 THEN 280
250 DI=0
260 DJ=-1
270 GOTO 310
280 IF K<>88 THEN 150
290 DI=1
300 DJ=0
310 CALL HCHAR(I,J,32)
320 I=I+DI
330 J=J+DJ
340 IF I>0 THEN 360
350 I=24
360 IF I<25 THEN 380
370 I=1
380 IF J>0 THEN 400
390 J=32
400 IF J<33 THEN 420
410 J=1
420 GOTO 140
430 STOP
```

The automatic "wraparound" coding can be made more compact by replacing lines 340-410 in the example above by:

```
340 I=INT(24*((I-1)/24-INT((I-1)/24)))+1
350 J=INT(32*((J-1)/32-INT((J-1)/32)))+1
```

These methods of using the arrow keys use CALL KEY (0,K,S), where 0 scans the whole keyboard so you can also detect if a space bar or ENTER key or period key has been pressed to fire. If you are only going to move and do not need a firing key, you may wish to use Keyboard 1 of the split keyboard. The arrow key codes for Keyboard 1 are 5 for up, 2 for left, 3 for right, and 0 for down. You could use an ON . . . GOTO statement rather than several IF statements. Since we do not want to recognize a key

press for codes 1 and 4, those GOTO line numbers will refer to the CALL KEY statement. Here is an example:

```
100 REM ARROW KEYS--3
110 CALL CLEAR
120 I=12
130 J=15
140 CALL HCHAR(I,J,64)
150 CALL KEY(1,K,S)
160 IF (K<0)+(K>5)THEN 150
170 ON K+1 GOTO 270,150,240,210,150,180
180 DI=-1
190 DJ=0
200 GOTO 290
210 DI=0
220 DJ=1
230 GOTO 290
240 DI=0
250 DJ=-1
260 GOTO 290
270 DI=1
280 DJ=0
290 CALL HCHAR(I,J,32)
300 I=I+DI
310 J=J+DJ
320 I=INT(24*((I-1)/24-INT((I-1)/24)))+1
330 J=INT(32*((J-1)/32-INT((J-1)/32)))+1
340 GOTO 140
350 STOP
```

Split Keyboard

A split keyboard approach is good when two players are using the arrow keys. The player at the left uses the regular arrow keys, and the player at the right uses the keys I, J, K, and M. To detect the arrow keys on the right half of the keyboard, use Keyboard 2 in the CALL KEY statement. The codes are the same for both sides of the keyboard: 5 for up, 2 for left, 3 for right, and 0 for down.

There may be a problem in testing for zero on the TI-99/4A console, so use logic such as IF K+1<>1 rather than IF K<>0.

Some of the key codes for the split keyboard on the TI-99/4 console are different from those on the TI-99/4A console. If you use the split keyboard scanning Keyboards 1 and 2 in the CALL

KEY statements, avoid using ENTER, G, B, slash, semicolon, comma, period, SHIFT, and the space bar.

When you have a two-player game, alternate between the CALL KEY statements for each half of the keyboard. In the previous listing, change line 340 to CALL HCHAR (I,J,64), then change line 350 to CALL KEY(2,K2,S2) and proceed with the second keyboard logic. Also change lines 160 and 170 to go to the second CALL KEY statement instead of returning to the first CALL KEY statement. After the second keyboard logic, return to the first keyboard logic. See the game "Labyrinth 2" for an illustration of this method.

Another programming method is using variables for the keyboard number, the coordinates, and the character number. Here is an example:

```
100 REM SPLIT KEYBOARD
110 CALL CLEAR
120 I(1)=8
130 J(1)=4
140 I(2)=8
150 J(2)=28
160 CALL HCHAR(8,4,65)
170 CALL HCHAR(8,28,66)
180 FOR C=1 TO 2
190 CALL KEY(C,K,S)
200 IF (K<0)+(K>5)THEN 390
210 ON K+1 GOTO 310,390,280,250,390,220
220 DI=-1
230 DJ=0
240 GOTO 330
250 DI=0
260 DJ=1
270 GOTO 330
280 DI=0
290 DJ=-1
300 GOTO 330
310 DI=1
320 DJ=0
330 CALL HCHAR(I(C),J(C),32)
340 I(C)=I(C)+DI
350 J(C)=J(C)+DJ
360 I(C)=INT(24*((I(C)-1)/24-INT((I(C)-1
```

```
    )/24)))+1
370 J(C)=INT(32*((J(C)-1)/32-INT((J(C)-1
    )/32)))+1
380 CALL HCHAR(I(C),J(C),64+C)
390 NEXT C
400 GOTO 180
410 END
```

Joysticks

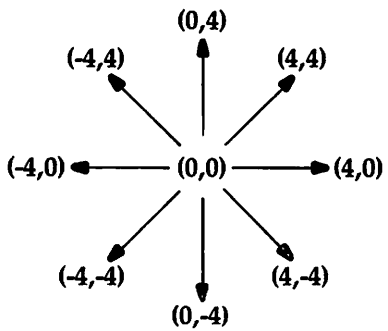
Wired Remote Controllers, or joysticks, as they are commonly called, are accessories available for your TI home computer. The TI controller consists of two joystick units connected to a single plug which can be inserted in the left side of the console. The joystick is operated by pressing the lever in the desired direction or by pressing the fire button. An instruction pamphlet is included with the joysticks.

Note: If you are using the TI-99/4A console with joysticks, be sure the ALPHA LOCK key is released (the key in the *up* position) or the lever on the joysticks will not operate correctly.

The joysticks in a program are controlled by the CALL JOYST command:

CALL JOYST *(unit,x,y)*

where *unit* may be 1 for joystick 1, or 2 for joystick 2, or a numeric variable which will equal 1 or 2. *x* and *y* are numeric variables. The computer will return an *x* and *y* value depending on the position of the joystick lever. The center position is (0,0), and the other positions have the value as shown in the figure.



Joystick Direction Values

The following program illustrates how you can draw using one joystick. A symbol appears in the middle of the screen; then as you move the joystick, the symbol leaves a trail in that direction. Line 140 detects the joystick position. Lines 150-160 update the coordinates for the symbol, and lines 170-180 allow the symbol to wrap to the other edge of the screen at the borders.

```
100 REM USING JOYSTICKS
110 CALL CLEAR
120 I=15
130 J=12
140 CALL JOYST(1,II,JJ)
150 I=I+II/4
160 J=J-JJ/4
170 I=INT(32*((I-1)/32-INT((I-1)/32)))+1
180 J=INT(24*((J-1)/24-INT((J-1)/24)))+1
190 CALL HCHAR(J,I,35)
200 GOTO 140
```

The fire button is detected with a CALL KEY statement. The form is

CALL KEY ***(unit, key, status)***

where unit is 1 or 2 for the joystick number. If the fire button is pressed, the key value returned is 18 (and 0 if the button is not pressed). You may prefer to check the status variable, which is 0 if the fire button is not pressed, −1 if the fire button is still pressed since the last CALL KEY, and +1 if the fire button was not pressed previously but now is pressed at this execution of the CALL KEY statement.

To see how the fire button is used in a program, add the following lines to the previous program. If the fire button is pressed, there will be a short beep.

```
142 CALL KEY(1,K,S)
144 IF K<>18 THEN 150
146 CALL SOUND(150,-2,4)
```

Joystick or Arrow Keys

Several methods of programming allow the player to use either the standard arrow keys or a joystick. This is one method:

```
100 REM KEYS OR JOYSTICKS
110 CALL CLEAR
120 I=15
130 J=12
140 CALL HCHAR(J,I,35)
150 CALL KEY(1,K,S)
160 IF S=0 THEN 290
170 IF K>5 THEN 150
180 II=0
190 JJ=0
200 ON K+1 GOTO 210,150,230,250,150,270
210 JJ=-4
220 GOTO 320
230 II=-4
240 GOTO 310
250 II=4
260 GOTO 310
270 JJ=4
280 GOTO 320
290 CALL JOYST(1,II,JJ)
300 IF (II=0)+(JJ=0)=-2 THEN 150
310 I=I+II/4
320 J=J-JJ/4
330 I=INT(32*((I-1)/32-INT((I-1)/32)))+1
340 J=INT(24*((J-1)/24-INT((J-1)/24)))+1
350 GOTO 140
```

Lines 150-280 detect if an arrow key on the keyboard is pressed. II and JJ are variables for the change in direction. Line 290 detects if the joystick lever is used. Line 160 tells the program to check the joystick if a key is not pressed. Line 300 tells the program to check the keyboard if the joystick lever is in the straight up (not touched) position. For either type of input, lines 310-340 determine the new coordinates for the symbol.

Another way to have both types of input in the same program is to ask at the beginning of the program whether the player wants to use joysticks or the keyboard (such as PRESS 1 FOR JOYSTICKS and PRESS 2 FOR KEYBOARD). Depending on the player's choice, the program branches to specific sections of code. The response time with this method can be slightly quicker because the previous method alternates polling the keyboard and joystick.

The fastest response time can be achieved by having two separate programs: one for joysticks, the other for keyboard. This way you do not have to check both joysticks and keyboard with each move, and the program will be shorter. (The longer the program, or the more memory it takes, the longer the response time for each move.)

Two Joysticks

Two joysticks may be used for two-player games. The programming technique is similar to the split-keyboard approach where players press two sets of arrow keys. Again, there are many ways to program the use of two joysticks. Essentially, you alternately check joystick 1 and joystick 2. Below is a sample program using two joysticks:

```
100 REM TWO JOYSTICKS
110 X(1)=10
120 Y(1)=12
130 X(2)=20
140 Y(2)=12
150 CALL CLEAR
160 CALL COLOR(3,7,7)
170 CALL COLOR(4,6,6)
180 FOR J=1 TO 2
190 CALL JOYST(J,DX,DY)
200 CALL KEY(J,K,S)
210 IF K<>18 THEN 230
220 CALL SOUND(100,-J,2)
230 CALL HCHAR(Y(J),X(J),32)
240 X(J)=X(J)+DX/4
250 Y(J)=Y(J)-DY/4
260 X(J)=INT(32*((X(J)-1)/32-INT((X(J)-1
    )/32)))+1
270 Y(J)=INT(24*((Y(J)-1)/24-INT((Y(J)-1
    )/24)))+1
280 CALL HCHAR(Y(J),X(J),54+J)
290 NEXT J
300 GOTO 180
```

Any variables with the subscript 1 pertain to the first joystick; any variables with the subscript 2 pertain to the second joystick. X(J) and Y(J) are the coordinates of the colored squares. Lines 180-300 first check joystick 1, then joystick 2, then repeat the process.

J is used as a variable for the unit number in the CALL JOYST and CALL KEY commands. Line 220 beeps noise 1 or noise 2 if the fire button is pressed. In your own game you could put procedures here to fire a bullet or whatever you wish when the fire button is pressed.

Random Numbers

Be sure to use the RANDOMIZE statement before you use the RND function so that each game you play will be different. If you compute random numbers several places in the game, you may need the RANDOMIZE statement before each RND to get better randomization throughout the game. Often a single RANDOMIZE at the beginning of the program does not help.

To obtain a random number from 1 to A, use this form:

R=INT(A*RND)+1 or **R=INT(RND*A+1)**

INT returns an integer, or whole number. RND actually returns a random number between 0 and 1 (a decimal number), so multiply it by the whole number you want, to get a random number from 0 to that number. Add 1 to get the random number from 1 to the number, including the number.

For a dice game you will need a random number from 1 to 6 for each throw of the die. Program 1 chooses the random number and then branches to the appropriate subroutine to draw the die.

For dealing a deck of cards, you can use the RND function. You may use an array to keep track of the cards. Since there are 52 cards, you could use an array C$(52), where each element is one of the cards. You may prefer to use a two-dimensional array such as C(13,4), where the first number indicates which number the card is and the second number indicates which of the four suits is chosen. Remember to use a DIMension statement for arrays using more than ten elements.

Program 2 illustrates how to deal five cards from a deck. The C array is DIMensioned to hold the 13 numbers and the 4 suits. Lines 130-270 define the graphics for the cards—you should use more detailed graphics for a fancier game. Line 290 selects N, the number of the card, and line 300 selects S, the suit of the card. All of the elements of the C array are initially zero, but as a card is chosen, C(N,S) is set equal to 1 (line 500). Line 310 makes sure the card has not previously been chosen. Lines 320-470 check the number of the card, and for 1, 11, 12, and 13 print A, J, Q, and K

instead of the numbers. Otherwise, the number is printed.

The remainder of the program asks if you want to deal five more cards. Option 1 assumes you cannot use cards over again. Option 2 assumes you put those five cards back in the deck and start anew.

Defining a function at the beginning of the program can be more efficient than typing a long formula for a random number each time it is needed. The following program draws random graphics. First, functions are defined. RI is a random row number from 1 to 24. RJ is a random column number from 2 to 30.

```
100 DEF RI=INT(RND*24)+1
110 DEF RJ=INT(RND*29)+2
120 CALL CLEAR
130 RANDOMIZE
140 FOR A=1 TO 20
150 CALL HCHAR(RI,RJ,42)
160 NEXT A
170 CALL HCHAR(RI,RJ,64)
180 PROGRAM WOULD CONTINUE HERE
```

Another use of random numbers is choosing from among a group of subroutines. For random subroutines:

```
200 ON INT(5*RND)+1 GOSUB 1000,2000,3000
    ,4000,5000
```

To pick a message, predefine an array of messages, then

```
700 PRINT M$(INT(10*RND))
```

You may wish to play random sounds:

```
900 CALL SOUND(150,INT(2000*RND+500),2)
```

Detecting Objects

There are several ways to determine the positions of objects on the screen so you can detect if you have crashed into the border or into another player or if you have reached your target. One way is to remember that the screen has 24 rows and 32 columns. Use the CALL GCHAR command to determine what character is at a certain position denoted by row and column numbers.

CALL GCHAR (X,Y,G) tells the computer to check row X and column Y and return the value of G, which is the character number (ASCII code) of the character in that position. You may

then branch depending on the value of G—if G = 32 there is a space at screen location X,Y; if G = 96, ASCII character is currently drawn there; if G = 104, a different kind of character appears in the screen location. Depending on G, you would move or score or crash—or whatever.

Another way of detecting objects is to keep track of the screen positions in an array such as S(24,32) for the 24 rows and 32 columns. This method is best used when the obstacles are stationary, such as a maze. You would test S(3,5) to find out the character in the third row and fifth column. For example:

```
300 IF S(3,5)=1 THEN 800
```

Timing

The TI-99/4A does not have a realtime clock, but you can simulate timing by incrementing a variable between CALL KEY statements. The timing will not be exactly even because when you press a key the loop slows down, but a timer can make a game more exciting. You may wish to actually print the time each loop (which slows down the action), or you can just increment the timer and then print the time at the end.

For example, this program counts the time until you press the ENTER key, then prints the results:

```
100 REM TIMING
110 CALL CLEAR
120 PRINT "PRESS ENTER"
130 CALL KEY(0,K,S)
140 T=T+1
150 IF K<>13 THEN 130
160 PRINT "TIME =";T
170 END
```

This program displays a timer which counts up until you press ENTER:

```
100 REM TIMING
110 CALL CLEAR
120 PRINT "PRESS ENTER"
130 PRINT :::"TIME ="
140 CALL KEY(0,K,S)
150 T=T+1
160 T$=STR$(T)
```

```
170 FOR I=1 TO LEN(T$)
180 CALL HCHAR(23,9+I,ASC(SEG$(T$,I,1)))
190 NEXT I
200 IF K<>13 THEN 140
210 CALL SOUND(150,1497,2)
220 END
```

If you want exact timing in your game, use the CALL SOUND statement and specify the exact number of milliseconds you need (1000 milliseconds = 1 second). For example, for a rocket countdown, CALL SOUND (1000,9999,30) for each second. The high frequency and softest volume make the sound undetectable.

The games in this book illustrate these and other programming techniques. You'll find some of these techniques useful in your own programming—and don't be afraid to experiment. It's the best way to learn.

Program 1. Dice

```
100 REM  DICE
110 CALL CHAR(128,"")
120 CALL CHAR(129,"3C7EFFFFFFFF7E3C")
130 CALL COLOR(13,2,16)
140 RANDOMIZE
150 CALL CLEAR
160 D=INT(6*RND+1)
170 FOR I=8 TO 16
180 CALL HCHAR(I,12,128,9)
190 NEXT I
200 ON D GOSUB 270,290,320,350,390,420
210 PRINT "TRY AGAIN? (Y/N)"
220 CALL KEY(0,K,S)
230 IF K=89 THEN 150
240 IF K<>78 THEN 220
250 CALL CLEAR
260 STOP
270 CALL HCHAR(12,16,129)
280 RETURN
290 CALL HCHAR(10,14,129)
300 CALL HCHAR(14,18,129)
310 RETURN
320 GOSUB 290
330 CALL HCHAR(12,16,129)
340 RETURN
350 GOSUB 290
360 CALL HCHAR(10,18,129)
```

```
370 CALL HCHAR(14,14,129)
380 RETURN
390 GOSUB 350
400 CALL HCHAR(12,16,129)
410 RETURN
420 GOSUB 350
430 CALL HCHAR(12,14,129)
440 CALL HCHAR(12,18,129)
450 RETURN
460 END
```

Program 2. Cards

```
100 REM  CARDS
110 CALL CLEAR
120 DIM C(13,4)
130 CALL CHAR(135,"6CFEFEFE7C7C381")
140 CALL CHAR(134,"10387CFE7C381")
150 CALL CHAR(130,"")
160 CALL COLOR(13,7,16)
170 CALL CHAR(136,"10387CEEFE7C1038")
180 CALL CHAR(137,"10381054FE541038")
190 CALL COLOR(14,2,16)
200 FOR I=3 TO 8
210 CALL COLOR(I,2,16)
220 NEXT I
230 CALL CHAR(92,"")
240 FOR I=1 TO 5
250 FOR J=3+(I-1)*6 TO 7+(I-1)*6
260 CALL VCHAR(5,J,130,7)
270 NEXT J
280 RANDOMIZE
290 N=INT(13*RND)+1
300 S=INT(4*RND)+1
310 IF C(N,S)=1 THEN 290
320 IF N<>10 THEN 360
330 CALL HCHAR(7,J-4,49)
340 CALL HCHAR(7,J-3,48)
350 GOTO 490
360 IF N<>11 THEN 390
370 CALL HCHAR(7,J-3,74)
380 GOTO 490
390 IF N<>12 THEN 420
400 CALL HCHAR(7,J-3,81)
410 GOTO 490
420 IF N<>13 THEN 450
430 CALL HCHAR(7,J-3,75)
440 GOTO 490
450 IF N<>1 THEN 480
```

```
460 CALL HCHAR(7,J-3,65)
470 GOTO 490
480 CALL HCHAR(7,J-3,N+48)
490 CALL HCHAR(9,J-3,133+S)
500 C(N,S)=1
510 NEXT I
520 CALL HCHAR(21,3,32,14)
530 CALL HCHAR(22,3,32,14)
540 PRINT "PRESS\1\DEAL\FIVE\MORE\\\\\\"
550 PRINT "\\\\\\\2\DEAL\FROM\FULL\DECK\"
560 PRINT "\\\\\\\3\END\PROGRAM\\\\\\\\\"
570 CALL KEY(0,K,ST)
580 IF (K<49)+(K>51)THEN 570
590 CALL CLEAR
600 ON K-48 GOTO 610,650,710
610 T=T+1
620 IF T<10 THEN 240
630 PRINT "OUT\OF\CARDS;\"
640 PRINT "STARTING\OVER\"::
650 FOR I=1 TO 13
660 FOR J=1 TO 4
670 C(I,J)=0
680 NEXT J
690 NEXT I
700 GOTO 240
710 END
```

1

Maze Generator

Charles Bond
Translated for the TI by C. Regena

Making the Maze

Here is a method of generating a maze with only one correct path. The maze must be a rectangle with dimensions of an odd number of squares. In the example, the rectangle is 27 squares by 19 squares with the upper-left corner in screen position 3,3.

The field or barriers are blue. The path is yellow and must start in a square with even coordinates. You may prefer to start in a particular square, such as the upper-left corner (4,4). The example program starts in a random square. Lines 170-180 set the starting coordinates.

From the starting square, a random number from 0 to 3 is chosen for the following possible directions:

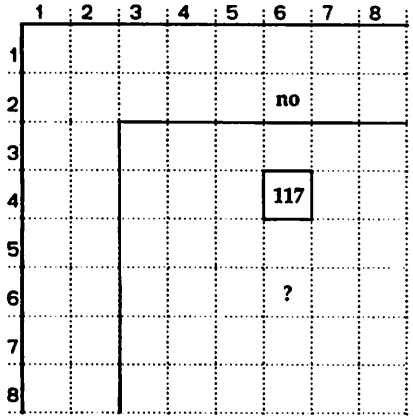
0 →
1 ↑
2 ←
3 ↓

The square *two* spots away from the starting square is tested. If it is a blue square, the maze path can go in that direction. A direction marker is placed in that square, and the square in between is made part of the path. If the square is off the field, the direction is unacceptable, and another direction will be chosen.

Plotting Each Square

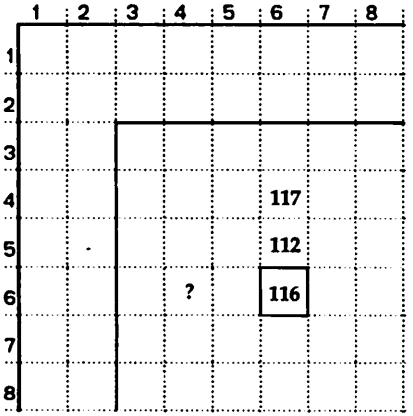
The example program, Program 1, starts in a random square. For illustration purposes, let's assume we start in square 4,6. (See Figure 1.) Place a yellow square there of character 117. A random direction is chosen—let's say 3, down. The square two down is blue, so the direction is acceptable. A yellow square 112 is drawn one down, and another yellow square 113 plus direction 3 = 116 is drawn two squares down.

Figure 1.
Starting the Maze



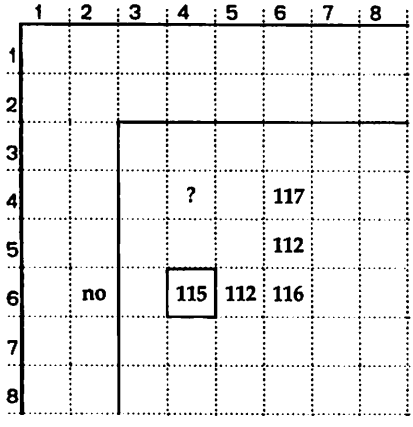
The new starting square is 6,6 (Figure 2). A random direction is chosen. The path cannot go up. Let's say 2, left, is chosen, which is acceptable.

Figure 2.
Moving Left



From this starting position the direction cannot be left (off the field) or right (116, already path). Assume 1, up, is chosen. (See Figure 3.)

Figure 3.
Moving Up



Now the position is 4,4. The path cannot move up or left because two squares away is off the field (Figure 4). The path cannot move to the right because two squares away is 117, the original starting position. The path must retrace and go down. The starting square this time is replaced by 112 (lines 430-460).

Figure 4.
A Dead End

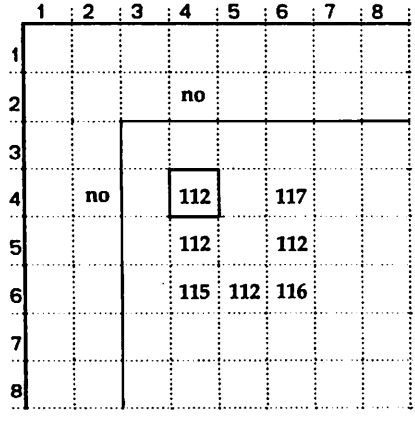
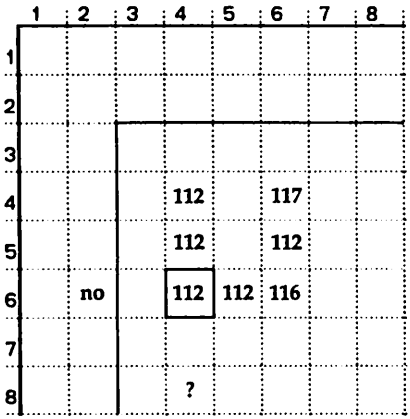


Figure 5. Backtracking

From this square the path can go down. Eventually, when the paths have been retraced to the beginning square, 117, the maze is finished.



To see the starting square for each step in the maze generator, add the following lines:

```
1Ø5 CALL COLOR(9,7,7)
385 CALL HCHAR(BX,BY,96)
455 CALL HCHAR(AX,AY,96)
```

Program Description

Line number	Explanation
100	Clear screen.
110	Define color for field (barriers).
120	Define color for path.
130-150	Draw blue field.
160	Randomize for RND function.
170-190	Designate starting square A (even numbers).
200-330	Choose random direction, right, up, left, or down.
340-360	Check square two squares away from A.
370	If not okay, go to 430.
380	Place path square in next spot.
390	Place direction marker two spots away from A.
400-410	New position becomes starting square.
420	Repeat process.

430-440	Choose different direction from 0 to 3. If direction is different, repeat drawing process.
450	Check direction marker.
460	Replace with yellow square 112.
470-550	Turn around (retrace path) and repeat process.
560	Play ending tone.
570	Hold maze on screen.

If you have the RS-232 interface and a printer, you can print the maze by adding the lines in Program 2. The blue field is replaced by the # symbol, and the path consists of spaces. Use your own printer configuration in line 570. See Figure 6 for an example of the maze produced with a printer.

Figure 6. Maze Produced with a Printer

```
############################
#       #    #             #    #
# ### # # # # ######### # # #
#   #   # # #   #     #   # #
# # ######### # ### ##### #
# #           #   #   # # #
# ############ ### # # # #
#   # # #         #   #   # #
### # # ######### ######## #
#   #   #     #   # #     #
# ### ##### ### ### # #####
#   #   #   #   #   # #   #
### ### # ### # # ### # # #
# #   #   #   # # #     # #
# ### ### # ### # ##### # #
#       #   # #   # #   # # #
# ######### # ### # # ### #
#           #   #   #     #
############################
```

Program 1. Maze

```
100 CALL CLEAR
110 CALL COLOR(10,5,5)
120 CALL COLOR(11,12,12)
130 FOR X=3 TO 21
```

```
140 CALL HCHAR(X,3,104,27)
150 NEXT X
160 RANDOMIZE
170 AY=2*(INT(12*RND))+4
180 AX=2*(INT(9*RND))+4
190 CALL HCHAR(AX,AY,117)
200 J=INT(RND*4)
210 JJ=J
220 ON J+1 GOTO 230,260,290,320
230 DX=0
240 DY=2
250 GOTO 340
260 DX=-2
270 DY=0
280 GOTO 340
290 DX=0
300 DY=-2
310 GOTO 340
320 DX=2
330 DY=0
340 BX=AX+DX
350 BY=AY+DY
360 CALL GCHAR(BX,BY,G)
370 IF G<>104 THEN 430
380 CALL HCHAR(AX+DX/2,AY+DY/2,112)
390 CALL HCHAR(BX,BY,J+113)
400 AX=BX
410 AY=BY
420 GOTO 200
430 J=(J+1)*-(J<3)
440 IF J<>JJ THEN 220
450 CALL GCHAR(AX,AY,G)
460 CALL HCHAR(AX,AY,112)
470 ON G-112 GOTO 480,500,520,540,560
480 AY=AY-2
490 GOTO 200
500 AX=AX+2
510 GOTO 200
520 AY=AY+2
530 GOTO 200
540 AX=AX-2
550 GOTO 200
560 CALL SOUND(150,1497,2)
570 GOTO 570
580 END
```

Program 2. Maze Printout

Add these lines for a printed copy of the maze.

```
570 OPEN #1:"RS232.BA=600"
580 FOR R=3 TO 21
590 FOR C=3 TO 29
600 CALL GCHAR(R,C,G)
610 IF G=112 THEN 640
620 PRINT #1:"#";
630 GOTO 650
640 PRINT #1:" ";
650 NEXT C
660 PRINT #1:CHR$(13)
670 NEXT R
680 CLOSE #1
690 GOTO 690
700 END
```

2

Maze Games

2

Hidden Maze

Gary Boden
Translated for the TI by C. Regena

The maze is there — you just can't see it! This maze game adds a new twist to maze puzzles.

Mazes present a challenge different from arcade-type "shoot-out" games, but the appeal of a maze can quickly fade once it has been solved. I have enhanced its challenge by hiding the complete maze from the player and showing only a realistically limited view from any position inside it. Although the view is from above rather than ground level, the player still gets a claustrophobic feeling similar to that of actually being inside the maze and groping along the corridors.

Playing Hidden Maze

"Hidden Maze" uses the method developed by Charles Bond (see "Maze Generator" in Chapter 1) to draw a maze on the screen—only the maze is hidden. It takes about two minutes to generate the maze. If you watch the blinking square you might get a hint of where the path of the maze is. You will then be shown a starting position with a limited view of the maze. Use the arrow keys to move—only a short distance of the maze will illuminate with each move. The objective is to get to the upper-left corner of the rectangle as quickly as possible.

Program Explanation

Lines 300 to 760 generate the maze. The path is generated with characters numbered 112-117, which at first are defined the same color as the background field so that the maze is hidden. Lines 790-850 uncover the maze around the starting position of Row 16 and Column 20. Character 136 is the "smiley face" marker that you move.

With each move, CALL GCHAR looks at surrounding squares and determines if each square should be part of the path or part of the barrier. The path squares are changed to yellow.

Line 780 initializes the time, T, to zero. Within each CALL KEY loop, line 910 increments T for the amount of time. The time

increments whether the character is moving or not—so thinking time adds to the time as well as the number of moves necessary to reach the upper-left corner.

Hidden Maze

```
100 CALL CLEAR
110 PRINT TAB(6);"***************"
120 PRINT TAB(6);"*";TAB(20);"*"
130 PRINT TAB(6);"* HIDDEN MAZE *"
140 PRINT TAB(6);"*";TAB(20);"*"
150 PRINT TAB(6);"***************"
160 PRINT :::"USE THE ARROW KEYS TO GET"
170 PRINT :"OUT OF THE MAZE."
180 PRINT ::"TRY TO GET TO THE UPPER"
190 PRINT :"LEFT CORNER IN THE LEAST"
200 PRINT :"AMOUNT OF TIME."
210 CALL COLOR(10,5,5)
220 CALL COLOR(11,5,5)
230 CALL COLOR(13,12,12)
240 CALL COLOR(14,7,12)
250 CALL CHAR(136,"3C7EDBFFDBE77E3C")
260 PRINT :::"PRESS ANY KEY TO BEGIN.";
270 CALL KEY(0,K,S)
280 IF S<1 THEN 270
290 CALL CLEAR
300 PRINT "GENERATING MAZE"
310 FOR X=3 TO 21
320 CALL HCHAR(X,3,104,27)
330 NEXT X
340 AY=4
350 AX=4
360 CALL HCHAR(AX,AY,117)
370 RANDOMIZE
380 J=INT(RND*4)
390 JJ=J
400 ON J+1 GOTO 410,440,470,500
410 DX=0
420 DY=2
430 GOTO 520
440 DX=-2
450 DY=0
460 GOTO 520
470 DX=0
480 DY=-2
490 GOTO 520
500 DX=2
510 DY=0
```

```
520 BX=AX+DX
530 BY=AY+DY
540 CALL GCHAR(BX,BY,G)
550 IF G<>104 THEN 620
560 CALL HCHAR(AX+DX/2,AY+DY/2,112)
570 CALL HCHAR(BX,BY,42)
580 CALL HCHAR(BX,BY,J+113)
590 AX=BX
600 AY=BY
610 GOTO 380
620 J=(J+1)*-(J<3)
630 IF J<>JJ THEN 400
640 CALL GCHAR(AX,AY,G)
650 CALL HCHAR(AX,AY,42)
660 CALL HCHAR(AX,AY,112)
670 ON G-112 GOTO 680,700,720,740,760
680 AY=AY-2
690 GOTO 380
700 AX=AX+2
710 GOTO 380
720 AY=AY+2
730 GOTO 380
740 AX=AX-2
750 GOTO 380
760 CALL SOUND(150,1497,2)
770 CALL HCHAR(23,3,32,15)
780 T=0
790 FOR A=15 TO 17
800 FOR B=19 TO 21
810 CALL GCHAR(A,B,G)
820 IF G=104 THEN 840
830 CALL HCHAR(A,B,128)
840 NEXT B
850 NEXT A
860 A=16
870 B=20
880 CALL HCHAR(A,B,136)
890 IF (A=4)+(B=4)=-2 THEN 1490
900 CALL KEY(1,K1,S)
910 T=T+1
920 IF (K1<0)+(K1>5)THEN 900
930 CALL HCHAR(A,B,128)
940 ON K1+1 GOTO 950,900,1100,1230,900,1360
950 CALL GCHAR(A+1,B,G)
960 IF G<>128 THEN 1080
970 CALL GCHAR(A+2,B-1,G)
980 IF G=104 THEN 1000
990 CALL HCHAR(A+2,B-1,128)
1000 CALL GCHAR(A+2,B,G)
```

```
1010 IF G=104 THEN 1030
1020 CALL HCHAR(A+2,B,128)
1030 CALL GCHAR(A+2,B+1,G)
1040 IF G=104 THEN 1060
1050 CALL HCHAR(A+2,B+1,128)
1060 A=A+1
1070 GOTO 880
1080 CALL SOUND(100,-5,4)
1090 GOTO 880
1100 CALL GCHAR(A,B-1,G)
1110 IF G<>128 THEN 1080
1120 CALL GCHAR(A,B-2,G)
1130 IF G=104 THEN 1150
1140 CALL HCHAR(A,B-2,128)
1150 CALL GCHAR(A-1,B-2,G)
1160 IF G=104 THEN 1180
1170 CALL HCHAR(A-1,B-2,128)
1180 CALL GCHAR(A+1,B-2,G)
1190 IF G=104 THEN 1210
1200 CALL HCHAR(A+1,B-2,128)
1210 B=B-1
1220 GOTO 880
1230 CALL GCHAR(A,B+1,G)
1240 IF G<>128 THEN 1080
1250 CALL GCHAR(A-1,B+2,G)
1260 IF G=104 THEN 1280
1270 CALL HCHAR(A-1,B+2,128)
1280 CALL GCHAR(A,B+2,G)
1290 IF G=104 THEN 1310
1300 CALL HCHAR(A,B+2,128)
1310 CALL GCHAR(A+1,B+2,G)
1320 IF G=104 THEN 1340
1330 CALL HCHAR(A+1,B+2,128)
1340 B=B+1
1350 GOTO 880
1360 CALL GCHAR(A-1,B,G)
1370 IF G<>128 THEN 1080
1380 CALL GCHAR(A-2,B-1,G)
1390 IF G=104 THEN 1410
1400 CALL HCHAR(A-2,B-1,128)
1410 CALL GCHAR(A-2,B,G)
1420 IF G=104 THEN 1440
1430 CALL HCHAR(A-2,B,128)
1440 CALL GCHAR(A-2,B+1,G)
1450 IF G=104 THEN 1470
1460 CALL HCHAR(A-2,B+1,128)
1470 A=A-1
1480 GOTO 880
1490 FOR I=1 TO 5
```

```
1500 CALL SOUND(-100,INT(RND*600)+700,4)
1510 CALL SCREEN(16)
1520 CALL SOUND(-100,INT(RND*600)+700,4)
1530 CALL SCREEN(11)
1540 CALL SOUND(-100,INT(RND*600)+700,4)
1550 CALL SCREEN(8)
1560 NEXT I
1570 PRINT "TIME =";T
1580 PRINT "TRY AGAIN? (Y/N)";
1590 CALL KEY(0,K,S)
1600 IF K=89 THEN 290
1610 IF K<>78 THEN 1590
1620 CALL CLEAR
1630 END
```

2

Labyrinth 1

C. Regena

You're caught in a cave with numerous passageways. "Labyrinth 1" tests your ability to find your way out of the cave.

"Labyrinth 1" is a maze game for one person. This game uses a method that may print an "impossible" maze—that is, you could be completely blocked in. After the labyrinth is printed, you have the option to print a different labyrinth in case you did get an impossible situation.

You are caught in the labyrinth at the left edge of the screen. The object of the game is to get to the opposite side in as few moves as possible. Use the arrow keys to move up, down, left, or right. If you get to a barrier and prefer to dig through it rather than go around, press ENTER, but it will cost 10 points. Your score is printed during the game.

Program Explanation

Character 97, a, is defined as a blank square; and character 98, b, is defined as a block. Lines 140-450 are subroutines that print lines of a's and b's—blank squares and blocks. You may use your own patterns of a's and b's as long as you have 28 total letters between the quote marks. Lines 490-530 print the maze by randomly choosing a number from 1 to 16, then using an ON . . . GOSUB to print the line of blanks and blocks.

The CALL GCHAR statement in line 940 determines what type of character is in the square you want to move to. If the square contains "a" or is blank, you can move. Line 1020 tests the column of the new position. If you are in column 30, the game ends.

The score is variable T and is incremented for each move (line 1030). The score is printed in lines 1040-1070. In the "Hidden Maze" game T is incremented right after the CALL KEY statement, so with each CALL KEY loop the time increments—whether a key has been pressed or not. But the method in this game counts only the actual moves or attempted moves.

Lines 1430-1490 are the procedure if ENTER is pressed to erase a barrier. There is a sound, then a flashing of the barrier

(alternating characters 112 and 97), then the square is replaced by a blank. The score T is incremented by 10 as a penalty.

"Labyrinth 1"and the next program, "Labyrinth 2," use the same method to create the maze. If you wish to save some typing for Labyrinth 2, simply type lines 140 to 570 of Labyrinth 1 first. SAVE these lines to disk or tape. Then finish typing Labyrinth 1 as printed. Don't forget lines 100-130. The directions for Labyrinth 2 will explain how to use the saved program lines.

Labyrinth 1

```
100 REM  LABYRINTH 1
110 REM
120 GOTO 1230
130 REM
140 PRINT "aaabaaabaaabaaabaaabaaabaaaa"
150 RETURN
160 PRINT "abbaaaaabab bbabaaaabaabbaaaa"
170 RETURN
180 PRINT "aaaabaaaaabbaaabaaabaaaabaaa"
190 RETURN
200 PRINT "abaaaabaaaaaaabaaaaabaaaabaa"
210 RETURN
220 PRINT "aaaaaaaabaaaabababaabbabbbaaa"
230 RETURN
240 PRINT "baaaabaaaabab bbaaabaaaaaabaaa"
250 RETURN
260 PRINT "baaabaabbbbaaaabaaabbaabaaaa"
270 RETURN
280 PRINT "abaaabaaaabaabbaaabaaabaaaab"
290 RETURN
300 PRINT "aababbbaaaaabaaaaaaabaaaaaba"
310 RETURN
320 PRINT "aabbaaaabbaaaaaabbaaabaaaaaa"
330 RETURN
340 PRINT "baaabbaaaababbaaaaaabaabaaaa"
350 RETURN
360 PRINT "bbbaaaaababbbaaaaabaaabaaaaa"
370 RETURN
380 PRINT "abaaabbaaaabbaaaaaaaabaaaaab"
390 RETURN
400 PRINT "aabaaaababaaaaaabaaabbbaaaba"
410 RETURN
420 PRINT "baaaaabbaaaabbbbaaabbaaaabba"
430 RETURN
440 PRINT "abababaaaaabbaaaabaaaabbbaaa"
450 RETURN
460 CALL CLEAR
```

```
470 CALL SCREEN(16)
480 CALL HCHAR(23,3,98,28)
490 FOR A=2 TO 20
500 RANDOMIZE
510 B=INT(16*RND)+1
520 ON B GOSUB 140,160,180,200,220,240,260,
    280,300,320,340,360,380,400,420,440
530 NEXT A
540 PRINT :::
550 CALL HCHAR(21,3,98,28)
560 CALL VCHAR(1,2,98,21)
570 CALL VCHAR(1,30,112,21)
580 CALL VCHAR(16,30,97,3)
590 C=INT(18*RND)+2
600 CALL HCHAR(C,3,113)
610 FOR A=1 TO 18
620 CALL HCHAR(24,3+A,ASC(SEG$("CHANGE MAZE
    ? (Y/N)",A,1)))
630 NEXT A
640 CALL KEY(0,K,S)
650 IF K=89 THEN 460
660 IF K<>78 THEN 640
670 CALL HCHAR(24,4,32,18)
680 C1=0
690 C2=0
700 C3=3
710 T=0
720 FOR A=1 TO 6
730 CALL HCHAR(23,3+A,ASC(SEG$("SCORE:",A,1
    )))
740 NEXT A
750 CALL SOUND(150,1400,2)
760 CALL KEY(0,K,S)
770 IF S<1 THEN 940
780 IF K=13 THEN 1430
790 IF K<>69 THEN 830
800 C1=-1
810 C2=0
820 GOTO 940
830 IF K<>68 THEN 870
840 C2=1
850 C1=0
860 GOTO 940
870 IF K<>88 THEN 910
880 C1=1
890 C2=0
900 GOTO 940
910 IF K<>83 THEN 940
920 C2=-1
```

```
930 C1=0
940 CALL GCHAR(C+C1,C3+C2,GC)
950 IF GC=97 THEN 980
960 CALL SOUND(-150,-6,0)
970 GOTO 1030
980 CALL HCHAR(C,C3,97)
990 C=C+C1
1000 C3=C3+C2
1010 CALL HCHAR(C,C3,113)
1020 IF C3=30 THEN 1090
1030 T=T+1
1040 T$=STR$(T)
1050 FOR A=1 TO LEN(T$)
1060 CALL HCHAR(23,10+A,ASC(SEG$(T$,A,1)))
1070 NEXT A
1080 GOTO 760
1090 CALL SOUND(100,262,2)
1100 CALL SOUND(100,330,2)
1110 CALL SOUND(100,392,2)
1120 CALL SOUND(300,523,2)
1130 IF BS<=T THEN 1150
1140 BS=T
1150 B$="BEST: "&STR$(BS)
1160 FOR A=1 TO LEN(B$)
1170 CALL HCHAR(23,18+A,ASC(SEG$(B$,A,1)))
1180 NEXT A
1190 PRINT "TRY AGAIN? (Y/N)";
1200 CALL KEY(0,K,S)
1210 IF K=89 THEN 460
1220 IF K=78 THEN 1500 ELSE 1200
1230 CALL CLEAR
1240 PRINT TAB(9);"LABYRINTH"
1250 PRINT :::"YOU ARE CAUGHT IN A"
1260 PRINT :"LABYRINTH.  GO OUT THE"
1270 PRINT :"OPPOSITE SIDE AS FAST AS"
1280 PRINT :"YOU CAN.  USE THE ARROW"
1290 PRINT :"KEYS TO MOVE."
1300 PRINT ::"IF YOU MUST DIG OUT, PRESS"
1310 PRINT :"<ENTER> BUT IT WILL COST"
1320 PRINT :"10 POINTS."
1330 CALL CHAR(97,"0")
1340 CALL CHAR(98,"FFFFFFFFFFFFFFFF")
1350 CALL COLOR(9,14,1)
1360 CALL CHAR(112,"FFFFFFFFFFFFFFFF")
1370 CALL CHAR(113,"383810FF10387CFE")
1380 CALL COLOR(11,11,1)
1390 BS=9999
1400 PRINT :::"PRESS ENTER TO START.";
1410 CALL KEY(0,K,S)
```

```
1420 IF K=13 THEN 460 ELSE 1410
1430 CALL SOUND(100,-5,2)
1440 FOR A=1 TO 3
1450 CALL HCHAR(C,C3+1,112)
1460 CALL HCHAR(C,C3+1,97)
1470 NEXT A
1480 T=T+10
1490 GOTO 1040
1500 CALL CLEAR
1510 END
```

Labyrinth 2

C. Regena

"Labyrinth 2" uses the split-keyboard technique to create a head-to-head game.

"Labyrinth 2" is like the previous game except it is a game for two players, who are lost at opposite ends of the labyrinth. The first one to reach the opposite side of the labyrinth wins. Once the labyrinth is printed, the option to change the maze is offered in case there is an "impossible" maze where a player is blocked in. The player on the left moves by pressing the regular arrow keys on keys E, S, D, and X, and the player on the right moves by pressing the I, J, K, and M keys.

Program Explanation

The labyrinth is printed using the same method as in "Labyrinth 1." In fact, if you want to save typing effort, type lines 140 to 570 from Labyrinth 1 and SAVE a copy of just those lines—then add lines 100-130 and continue typing from line 580 for the Labyrinth 1 game. To type in Labyrinth 2, load your previously saved lines 140-570. Type RES 170 to resequence the lines starting with line 170. Now add lines 100-160 of Labyrinth 2, then continue typing from line 600 for the rest of the game.

The split-keyboard CALL KEY method is used to read the arrow keys for each player. The computer checks each player's moves alternately.

Labyrinth 2

```
100 REM  LABYRINTH 2
110 REM
120 GOTO 1940
130 FOR A=1 TO LEN(A$)
140 CALL HCHAR(B,A,ASC(SEG$(A$,A,1)))
150 NEXT A
160 RETURN
170 PRINT "aaabaaabaaabaaabaaabaaabaaaa"
180 RETURN
190 PRINT "abbaaaaabab bbabaaaabaabbaaaa"
200 RETURN
```

```
210 PRINT "aaaabaaaaabbaaabaaabaaaabaaa"
220 RETURN
230 PRINT "abaaaabaaaaaaabaaaaabaaaabaa"
240 RETURN
250 PRINT "aaaaaaaabaaaabababaabbabbbaaa"
260 RETURN
270 PRINT "baaaabaaaababbaaabaaaaaabaaa"
280 RETURN
290 PRINT "baaabaabbbbaaaabaaabbaabaaaa"
300 RETURN
310 PRINT "abaaabaaaabaabbaaabaaabaaaab"
320 RETURN
330 PRINT "aababbbaaaaabaaaaaaabaaaaaba"
340 RETURN
350 PRINT "aabbaaaabbaaaaaabbaaabaaaaaa"
360 RETURN
370 PRINT "baaabbaaaababbaaaaaabaabaaaa"
380 RETURN
390 PRINT "bbbaaaaabababbaaaaabaaabaaaa"
400 RETURN
410 PRINT "abaaabbaaaabbaaaaaaaabaaaaab"
420 RETURN
430 PRINT "aabaaaabababaaaaaabaaabbbaaba"
440 RETURN
450 PRINT "baaaaabbaaaabbbbaaabbaaaabba"
460 RETURN
470 PRINT "abababaaaaabbaaaabaaaabbbaaa"
480 RETURN
490 CALL CLEAR
500 CALL HCHAR(23,3,98,28)
510 FOR A=2 TO 20
520 RANDOMIZE
530 B=INT(16*RND)+1
540 ON B GOSUB 170,190,210,230,250,270,290,
    310,330,350,370,390,410,430,450,470
550 NEXT A
560 PRINT :::
570 CALL HCHAR(21,3,98,28)
580 CALL VCHAR(1,2,104,21)
590 CALL VCHAR(1,30,112,21)
600 CALL VCHAR(4,2,97,3)
610 CALL VCHAR(16,30,97,3)
620 C=INT(18*RND)+2
630 CALL HCHAR(C,3,113)
640 D=INT(18*RND)+2
650 CALL HCHAR(D,29,105)
660 A$="CHANGE MAZE? (Y/N)"
670 B=24
680 GOSUB 130
```

```
690 CALL KEY(0,K,S)
700 IF K=89 THEN 490
710 IF K<>78 THEN 690
720 CALL HCHAR(24,1,32,19)
730 C1=0
740 C2=0
750 D1=0
760 D2=0
770 C3=3
780 D3=29
790 CS=0
800 DS=0
810 CALL SOUND(150,1400,2)
820 CALL KEY(1,K1,S1)
830 IF S1=0 THEN 990
840 IF K1<>5 THEN 880
850 C1=-1
860 C2=0
870 GOTO 990
880 IF K1<>3 THEN 920
890 C2=1
900 C1=0
910 GOTO 990
920 IF K1+1<>1 THEN 960
930 C1=1
940 C2=0
950 GOTO 990
960 IF K1<>2 THEN 990
970 C2=-1
980 C1=0
990 IF C3+C2>1 THEN 1030
1000 A$="RED WENT WRONG WAY-BLUE WON."
1010 DS=1
1020 GOTO 1350
1030 CALL GCHAR(C+C1,C3+C2,GC)
1040 IF GC=97 THEN 1080
1050 IF GC=105 THEN 1510
1060 CALL SOUND(-150,-6,0)
1070 GOTO 1150
1080 CALL HCHAR(C,C3,97)
1090 C=C+C1
1100 C3=C3+C2
1110 CALL HCHAR(C,C3,113)
1120 IF C3<>30 THEN 1150
1130 CS=1
1140 GOTO 1630
1150 CALL KEY(2,K2,S2)
1160 IF S2=0 THEN 1320
1170 IF K2<>5 THEN 1210
```

```
1180 D1=-1
1190 D2=0
1200 GOTO 1320
1210 IF K2<>3 THEN 1250
1220 D2=1
1230 D1=0
1240 GOTO 1320
1250 IF K2+1<>1 THEN 1290
1260 D1=1
1270 D2=0
1280 GOTO 1320
1290 IF K2<>2 THEN 1320
1300 D2=-1
1310 D1=0
1320 IF D3+D2<31 THEN 1390
1330 A$="BLUE WENT WRONG WAY-RED WON."
1340 CS=1
1350 B=22
1360 CALL SOUND(200,-8,2)
1370 GOSUB 130
1380 GOTO 1740
1390 CALL GCHAR(D+D1,D3+D2,GD)
1400 IF GD=97 THEN 1440
1410 IF GD=113 THEN 1510
1420 CALL SOUND(-150,-6,0)
1430 GOTO 820
1440 CALL HCHAR(D,D3,97)
1450 D=D+D1
1460 D3=D3+D2
1470 CALL HCHAR(D,D3,105)
1480 IF D3<>2 THEN 820
1490 DS=1
1500 GOTO 1630
1510 CALL HCHAR(D,D3,97)
1520 CALL HCHAR(C,C3,120,2)
1530 CALL HCHAR(C+C1,C3+C2,120,2)
1540 CALL SOUND(300,-8,2)
1550 FOR A=1 TO 15
1560 CALL COLOR(12,7,12)
1570 CALL COLOR(12,12,7)
1580 NEXT A
1590 A$="CRASH!!  NO WINNER."
1600 B=23
1610 GOSUB 130
1620 GOTO 1900
1630 CALL HCHAR(23,1,32,64)
1640 IF DS=1 THEN 1670
1650 A$="RED WON!"
1660 GOTO 1680
```

```
1670 A$="BLUE WON!"
1680 B=22
1690 CALL SOUND(100,262,2)
1700 CALL SOUND(100,330,2)
1710 CALL SOUND(100,392,2)
1720 CALL SOUND(300,523,2)
1730 GOSUB 130
1740 A$="TOTAL SCORE:  RED{6 SPACES}BLUE "
1750 B=23
1760 GOSUB 130
1770 TCS=TCS+CS
1780 TDS=TDS+DS
1790 IF TCS<10 THEN 1830
1800 CALL HCHAR(23,19,49)
1810 CALL HCHAR(23,20,48)
1820 GOTO 1880
1830 CALL HCHAR(23,19,TCS+48)
1840 IF TDS<10 THEN 1880
1850 CALL HCHAR(23,29,49)
1860 CALL HCHAR(23,30,48)
1870 GOTO 2150
1880 CALL HCHAR(23,29,TDS+48)
1890 IF TCS=10 THEN 2150
1900 PRINT "TRY AGAIN? (Y/N)";
1910 CALL KEY(0,K,S)
1920 IF K=89 THEN 490
1930 IF K=78 THEN 2140 ELSE 1910
1940 CALL CLEAR
1950 PRINT TAB(9);"LABYRINTH"
1960 PRINT :::"TWO GIRLS ARE CAUGHT IN A"
1970 PRINT :"LABYRINTH.  THEY WILL RACE"
1980 PRINT :"TO GET OUT THE OPPOSITE"
1990 PRINT :"BORDERS.  USE THE ARROW"
2000 PRINT :"KEYS TO MOVE."
2010 CALL CHAR(97,"0")
2020 CALL CHAR(98,"FFFFFFFFFFFFFFFF")
2030 CALL COLOR(9,3,1)
2040 CALL CHAR(112,"FFFFFFFFFFFFFFFF")
2050 CALL CHAR(113,"383810FF10387CFE")
2060 CALL COLOR(11,7,1)
2070 CALL CHAR(120,"49221449142249")
2080 CALL CHAR(104,"FFFFFFFFFFFFFFFF")
2090 CALL CHAR(105,"383810FF10387CFE")
2100 CALL COLOR(10,6,1)
2110 PRINT :::"PRESS ENTER TO START."
2120 CALL KEY(0,K,S)
2130 IF K=13 THEN 490 ELSE 2120
2140 CALL CLEAR
2150 END
```

3

Chase Games

3

Superchase

Anthony Godshall
Translated for the TI by C. Regena

This chase game involves more than just staying ahead of the enemy. Good strategy can increase your score and prevent capture.

"Superchase" is an arcade-style game in which you try to find all the treasures before the enemy catches you. Sounds easy, doesn't it? Well, it isn't quite that simple.

Using the arrow keys to move left, right, up, or down in the maze, stay on the yellow path and try to pick up as many treasures (purple) as you can before the enemy chaser gets you. You leave a white trail, so the chaser knows where to go. Since the chaser will follow your exact trail, you can probably mix him up at the intersections and dead ends. After you pick up 45 treasures on a screen, the screen clears for a new maze and more treasures.

Program Explanation

Superchase presents a third method to draw a maze. Lines 320-360 draw the blue background. Lines 380-470 draw 65 random horizontal path segments and 60 random vertical path segments. Each path segment is five squares long and must start in a square with even coordinates. Lines 480-490 make sure the starting square is connected to the rest of the maze. This method of drawing a maze can leave segments that are impossible to get to. Sixty-five horizontal and 60 vertical path segments are used to minimize the chance of stray segments and to draw enough paths to made 45 treasures possible.

Treasures (character 96) are drawn randomly with lines 500-540. Character 120 is you, and character 128 represents the white trail character.

The variable T represents the number of treasures picked up. LV is the level number (which screen). Line 680 determines how many treasures are picked up before the chase begins.

Lines 1410-1730 move the players depending on the arrow key pressed. Lines 1460-1530 are used only when a treasure is collected. There is a short sound; the number of treasures, T, for the screen and the total score, SC, are incremented; and the score

is printed. When a barrier is hit the procedure at lines 1740-1750 is used. Lines 1760-1820 are the procedure if you get caught. Lines 1830-1880 present the option to try again. The chase procedure is found in lines 690-1400. As you get to more screens, fewer treasures need to be picked up before the superchase begins. The enemy chaser follows the white trail whenever possible. You may notice that, at an intersection, the chaser checks the choices in a certain order every time—so you can plan your player strategy accordingly. (If you prefer, you can change the program to choose randomly which direction to go.) PG is the previous graphic character number stored so the computer remembers what to draw as the chaser moves on. FL is a flag to try to prevent the chaser from getting stuck in a dead end.

Superchase

```
100 CALL CLEAR
110 PRINT TAB(6);"***************"
120 PRINT TAB(6);"* SUPERCHASE *"
130 PRINT TAB(6);"***************"
140 CALL CHAR(96,"00183C7E7E3C18")
150 CALL COLOR(9,14,12)
160 PRINT ::::"USE THE ARROW KEYS TO MOVE."
170 PRINT ::"TRY TO GATHER AS MANY"
180 PRINT :"TREASURES (`) IN THE MAZE"
190 PRINT :"AS POSSIBLE BEFORE THE"
200 PRINT :"ENEMY CATCHES YOU!"
210 CALL COLOR(10,5,5)
220 CALL COLOR(11,12,12)
230 CALL CHAR(120,"383810FE10387CFE")
240 CALL COLOR(12,7,12)
250 CALL CHAR(128,"383810FE10387CFE")
260 CALL COLOR(13,16,12)
270 PRINT :::"PRESS ANY KEY TO BEGIN.";
280 CALL KEY(0,K,S)
290 IF S<1 THEN 280
300 SC=0
310 LV=1
320 CALL CLEAR
330 CALL SCREEN(4)
340 FOR X=3 TO 21
350 CALL HCHAR(X,3,104,27)
360 NEXT X
370 RANDOMIZE
380 FOR I=1 TO 65
390 X=2*(INT(9*RND))+4
400 Y=2*(INT(11*RND))+4
```

```
410 CALL HCHAR(X,Y,112,5)
420 NEXT I
430 FOR I=1 TO 60
440 X=2*(INT(7*RND))+4
450 Y=2*(INT(13*RND))+4
460 CALL VCHAR(X,Y,112,5)
470 NEXT I
480 CALL HCHAR(4,4,112,5)
490 CALL VCHAR(4,4,112,5)
500 FOR I=1 TO 80
510 X=2*(INT(9*RND))+4
520 Y=2*(INT(13*RND))+4
530 CALL HCHAR(X,Y,96)
540 NEXT I
550 FOR I=1 TO 6
560 CALL HCHAR(23,2+I,ASC(SEG$("SCORE:",I,1)))
570 NEXT I
580 CALL SOUND(100,1497,2)
590 T=0
600 IF LV<8 THEN 620
610 LV=7
620 A=4
630 B=4
640 C=4
650 D=4
660 PG=112
670 CALL HCHAR(A,B,120)
680 IF T<8-LV THEN 1410
690 IF FL=0 THEN 740
700 IF FL/2=INT(FL/2)THEN 1140
710 CCX=CX
720 CCY=CY
730 FL=0
740 CALL GCHAR(C+1,D,GC)
750 IF GC=120 THEN 1760
760 IF GC=104 THEN 810
770 IF GC<>128 THEN 810
780 CX=1
790 CY=0
800 GOTO 1340
810 CALL GCHAR(C,D+1,GC)
820 IF GC=120 THEN 1760
830 IF GC=104 THEN 880
840 IF GC<>128 THEN 880
850 CX=0
860 CY=1
870 GOTO 1340
880 CALL GCHAR(C-1,D,GC)
890 IF GC=120 THEN 1760
```

```
900 IF GC=104 THEN 950
910 IF GC<>128 THEN 950
920 CX=-1
930 CY=0
940 GOTO 1340
950 CALL GCHAR(C,D-1,GC)
960 IF GC=120 THEN 1760
970 IF GC=128 THEN 1320
980 IF GC=112 THEN 1320
990 FL=FL+1
1000 IF FL/2=INT(FL/2)THEN 1040
1010 CX=CCX
1020 CY=CCY
1030 GOTO 1160
1040 CX=SGN(A-C)
1050 CY=0
1060 IF CX<>0 THEN 1080
1070 CY=SGN(D-B)
1080 CALL GCHAR(C+CX,D+CY,GC)
1090 IF GC=104 THEN 1110
1100 IF (GC=96)+(GC=112)THEN 1340
1110 CX=-1
1120 CY=0
1130 GOTO 1160
1140 FL=FL+1
1150 IF FL>1 THEN 1300
1160 CALL GCHAR(C+CX,D+CY,GC)
1170 IF (GC=112)+(GC=96)+(GC=128)THEN 1340
1180 CX=1
1190 CY=0
1200 CALL GCHAR(C+CX,D,GC)
1210 IF (GC=112)+(GC=96)+(GC=128)THEN 1340
1220 CX=-1
1230 CALL GCHAR(C+CX,D,GC)
1240 IF (GC=112)+(GC=96)+(GC=128)THEN 1340
1250 CX=0
1260 CY=-1
1270 CALL GCHAR(C,D+CY,GC)
1280 IF (GC=112)+(GC=96)+(GC=128)THEN 1340
1290 CY=1
1300 CALL GCHAR(C+CX,D+CY,GC)
1310 IF GC=104 THEN 1390 ELSE 1340
1320 CX=0
1330 CY=-1
1340 IF PG=96 THEN 1360
1350 PG=112+FL
1360 CALL HCHAR(C,D,PG)
1370 C=C+CX
1380 D=D+CY
```

```
1390 CALL HCHAR(C,D,42)
1400 PG=GC
1410 CALL KEY(1,K1,S)
1420 IF (K1<0)+(K1>5)THEN 1410
1430 CALL HCHAR(A,B,128)
1440 ON K1+1 GOTO 1540,1410,1590,1640,1410,1690
1450 IF G<>96 THEN 670
1460 CALL SOUND(100,-1,4)
1470 SC=SC+1
1480 SC$=STR$(SC)
1490 FOR I=1 TO LEN(SC$)
1500 CALL HCHAR(23,10+I,ASC(SEG$(SC$,I,1)))
1510 NEXT I
1520 T=T+1
1530 IF T<45 THEN 670 ELSE 320
1540 CALL GCHAR(A+1,B,G)
1550 IF G=104 THEN 1740
1560 IF G=42 THEN 1760
1570 A=A+1
1580 GOTO 1450
1590 CALL GCHAR(A,B-1,G)
1600 IF G=104 THEN 1740
1610 IF G=42 THEN 1760
1620 B=B-1
1630 GOTO 1450
1640 CALL GCHAR(A,B+1,G)
1650 IF G=104 THEN 1740
1660 IF G=42 THEN 1760
1670 B=B+1
1680 GOTO 1450
1690 CALL GCHAR(A-1,B,G)
1700 IF G=104 THEN 1740
1710 IF G=42 THEN 1760
1720 A=A-1
1730 GOTO 1450
1740 CALL SOUND(-100,-5,4)
1750 GOTO 670
1760 CALL SOUND(200,-6,4)
1770 FOR I=1 TO 3
1780 CALL SCREEN(16)
1790 CALL SCREEN(9)
1800 CALL SCREEN(8)
1810 NEXT I
1820 PRINT "GOT CAUGHT!!!"
1830 PRINT "TRY AGAIN? (Y/N)";
1840 CALL KEY(0,K,S)
1850 IF K=89 THEN 300
1860 IF K<>78 THEN 1840
1870 CALL CLEAR
1880 END
```

3

Marble Hunt

Ronny Ong
Translated for the TI by C. Regena

"Marble Hunt" is a unique kind of maze game. It has no maze! Without guiding corridors, the player must exercise control while keeping up with the fast pace of the game.

You are the red player on the playground. The object is to try to pick up as many marbles as you can, but watch out for the bully! If the bully catches you, the game is over. You have a slight advantage because your player can wrap around to the opposite edge of the screen, but the bully cannot, so you should plan your strategy accordingly.

Program Explanation

Your player is character number 96, and the bully is character number 104. The marbles are character number 128 and are a random color (lines 300-340). You always start the game at the upper-left corner (X = 1, Y = 3), but the bully starts in a random position (lines 350-360 and 410).

CALL KEY (1,K,S) is used to detect which arrow key you press, and only arrow keys are accepted. Lines 570-600 determine your new position after you press an arrow key. If you do not press an arrow key, the bully will catch up with you more quickly, so keep on the move.

The bully's chase procedure is in lines 680-810. The SGN function is used for a -1, 0,or +1 increment in direction depending on the bully's present position and your position.

Marble Hunt

```
100 CALL CLEAR
110 PRINT TAB(4);"***************"
120 PRINT TAB(4);"* MARBLE HUNT *"
130 PRINT TAB(4);"***************"
140 PRINT :::"USE THE ARROW KEYS TO"
150 PRINT :"MOVE AROUND TO PICK UP"
160 PRINT :"ALL THE MARBLES ON THE"
170 PRINT :"PLAYGROUND."
180 PRINT ::"WATCH OUT FOR THE BULLY!":::
```

```
190 CALL CHAR(128,"0000183C3C18")
200 CALL CHAR(96,"001C1C083E081414")
210 CALL COLOR(9,7,1)
220 CALL CHAR(104,"1C1C083E08141414")
230 PRINT "PRESS ANY KEY TO START.";
240 CALL KEY(0,K,S)
250 IF S<1 THEN 240
260 T=0
270 CALL CLEAR
280 X=1
290 Y=3
300 RANDOMIZE
310 C=INT(15*RND+2)
320 IF C=4 THEN 310
330 CALL COLOR(13,C,1)
340 CALL HCHAR(1,1,128,768)
350 A=INT(24*RND+1)
360 B=INT(25*RND+5)
370 PG=128
380 CALL SOUND(150,1497,2)
390 CALL HCHAR(X,Y,96)
400 CALL SOUND(150,110,2)
410 CALL HCHAR(A,B,104)
420 CALL KEY(1,K,S)
430 IF (K<0)+(K>5)THEN 680
440 ON K+1 GOTO 450,680,480,510,680,540
450 DX=1
460 DY=0
470 GOTO 560
480 DX=0
490 DY=-1
500 GOTO 560
510 DX=0
520 DY=1
530 GOTO 560
540 DX=-1
550 DY=0
560 CALL HCHAR(X,Y,32)
570 X=X+DX
580 Y=Y+DY
590 X=INT(24*((X-1)/24-INT((X-1)/24)))+1
600 Y=INT(32*((Y-1)/32-INT((Y-1)/32)))+1
610 CALL GCHAR(X,Y,G)
620 IF G=104 THEN 830
630 IF G=32 THEN 670
640 CALL SOUND(-100,1497,2)
650 T=T+1
660 IF T=767 THEN 870
670 CALL HCHAR(X,Y,96)
```

```
680 CALL HCHAR(A,B,PG)
690 A=A-SGN(A-X)
700 IF A>0 THEN 720
710 A=1
720 IF A<25 THEN 740
730 A=24
740 B=B-SGN(B-Y)
750 IF B>0 THEN 770
760 B=1
770 IF B<32 THEN 790
780 B=32
790 CALL GCHAR(A,B,PG)
800 IF PG=96 THEN 830
810 CALL HCHAR(A,B,104)
820 GOTO 420
830 CALL SOUND(300,-5,2)
840 PRINT "THE BULLY GOT YOU!!"
850 PRINT :"YOU GOT";T;"MARBLES."
860 GOTO 890
870 PRINT "CONGRATULATIONS!!"
880 PRINT "YOU GOT ALL THE MARBLES!!"
890 PRINT :::"TRY AGAIN? (Y/N)";
900 CALL KEY(0,K,S)
910 IF K=89 THEN 260
920 IF K<>78 THEN 900
930 PRINT :::::::
940 END
```

3

Closeout

L. L. Beh
Translated for the TI by C. Regena

"Closeout" uses the concept of artificial intelligence to create an interesting chase game.

There's a huge sale going on at a local department store. You arrive at the multistory building hungry for bargains. Boldly, you enter the store, look around, and see bargains galore. A real sale! Gathering up sale items, you suddenly become aware of two other shoppers. Wherever you go, they follow you. Soon you learn their true intentions—they are out to stop you at all costs so they can have the store to themselves.

The object of "Closeout" is to snatch up as many sale items as possible while evading the hostile bargain hunters. Don't let them get too close—if they capture you, the game is over.

You start on the ground floor at the bottom-left corner. Use the arrow keys to move left, right, up, or down. You may go up or down only at the escalators or at the elevators at the ends of the floors. Pick up the sale items by moving your character over the colored dots. If a fellow shopper gets too aggressive, you can deliver a shove that will send him reeling up to the top floor simply by pressing the space bar. But when you do, you lose 25 sale items. You start out with three shoves, and with each screen you clear you earn another one. Shoves can't be used on the escalators or elevators though.

Program Explanation

Characters 96 and 104 are the graphics characters to draw the floors and sale items. Lines 260-330 define A$ and B$ to be printed in line 520. The sale items are a different random color for each screen (line 500).

R is the number of rounds or screens; SC is the score or the number of sale items picked up minus the 25 points per shove used; and SH is the number of shoves available. Lines 460 and 470 increment R and SH for each round. T is the number of sale items picked up per screen, and when T = 180 the screen changes.

Lines 560-740 draw the escalators in random positions,

making sure that no two are right next to each other and that they are properly aligned on the floors.

All variables starting with E pertain to enemy shoppers. The subscripts 0 and 1 relate to the two shoppers. The SGN function returns −1 for a negative value, 0 for zero, and 1 for a positive value. These numbers then determine the direction in which the enemies chase your shopper. At any escalator or elevator the enemies attempt to get to your floor, then choose left or right depending on whether you are on the left or right. However, once the enemies start across the floor in a direction, they keep going in that direction even if you change. Therefore, plan your moves to stay away from the enemies.

The game may be speeded up by changing the program to have only one enemy shopper.

Closeout

```
100 REM  CLOSEOUT
110 CALL CLEAR
120 PRINT TAB(7);"************"
130 PRINT TAB(7);"* CLOSEOUT *"
140 PRINT TAB(7);"************"
150 PRINT :::"YOU ARE THE RED SHOPPER."
160 PRINT :"USE ARROW KEYS TO MOVE."
170 PRINT :"USE SPACE BAR TO SHOVE."
180 PRINT :"A SHOVE COSTS YOU 25 POINTS.":: 
    :::
190 CALL CHAR(96,"0000FFFFFFFF")
200 CALL COLOR(9,5,1)
210 CALL CHAR(104,"0000001818")
220 CALL CHAR(112,"383810FE10282828")
230 CALL COLOR(11,7,1)
240 CALL CHAR(120,"383810FE10282828")
250 CALL COLOR(13,3,1)
260 A$=CHR$(96)
270 B$=CHR$(104)
280 FOR I=1 TO 5
290 A$=A$&A$
300 B$=B$&B$
310 NEXT I
320 A$=SEG$(A$,1,28)
330 B$=SEG$(B$,1,28)
340 PRINT "PRESS <ENTER>"
350 CALL KEY(0,K,S)
360 IF K<>13 THEN 350
370 R=0
380 SC=0
```

```
390 SH=2
400 FOR L=0 TO 1
410 E(L,0)=6
420 EG(L)=104
430 EX(L)=0
440 NEXT L
450 CALL CLEAR
460 R=R+1
470 SH=SH+1
480 T=0
490 RANDOMIZE
500 CALL COLOR(10,INT(RND*12)+5,1)
510 PRINT "CLOSEOUT";TAB(19);"ROUND";R::
520 PRINT A$:B$:A$:B$:A$:B$:A$:B$:A$:B$:A$:
    B$:A$:B$:A$;B$:A$
530 PRINT :"SCORE:";SC,"SHOVES:";SH
540 CALL VCHAR(6,3,45,15)
550 CALL VCHAR(6,30,45,15)
560 FOR I=1 TO 7
570 R1(I)=INT(RND*23)+5
580 FOR J=1 TO I-1
590 IF ABS(R1(J)-R1(I))<2 THEN 570
600 NEXT J
610 R2=(INT(RND*6)+1)*2+6
620 IF R<3 THEN 660
630 R2=R2+2
640 IF R2<=18 THEN 660
650 R2=18
660 IF R2<18 THEN 690
670 R3=3
680 GOTO 730
690 IF R2<16 THEN 720
700 R3=2*(INT(RND*2)+1)+1
710 GOTO 730
720 R3=2*(INT(RND*3)+1)+1
730 CALL VCHAR(R2,R1(I),45,R3)
740 NEXT I
750 X=20
760 Y=4
770 A=32
780 CALL HCHAR(X,Y,112)
790 E(0,1)=INT(RND*8)+4
800 CALL HCHAR(E(0,0),E(0,1),120)
810 E(1,1)=INT(RND*8)+18
820 CALL HCHAR(E(1,0),E(1,1),120)
830 EY(0)=-1
840 EY(1)=1
850 L=0
860 CALL KEY(0,K,S)
```

```
870 IF K=32 THEN 1670
880 IF K<>69 THEN 930
890 IF X<=4 THEN 1060
900 DX=-1
910 DY=0
920 GOTO 1060
930 IF K<>68 THEN 980
940 IF Y>=29 THEN 1060
950 DX=0
960 DY=1
970 GOTO 1060
980 IF K<>83 THEN 1020
990 IF Y=3 THEN 1060
1000 DX=0
1010 DY=-1
1020 IF K<>88 THEN 1060
1030 IF X=20 THEN 1060
1040 DX=1
1050 DY=0
1060 CALL GCHAR(X+DX,Y+DY,G)
1070 IF G=120 THEN 1920
1080 IF (G<>45)+(G<>104)+(G<>32)=-3 THEN 12
     50
1090 CALL HCHAR(X,Y,A)
1100 X=X+DX
1110 Y=Y+DY
1120 CALL HCHAR(X,Y,112)
1130 IF (G=45)+(G=32)THEN 1240
1140 T=T+1
1150 IF T=180 THEN 400
1160 SC=SC+1
1170 A=32
1180 CALL SOUND(100,-5,2)
1190 SC$=STR$(SC)&"  "
1200 FOR I=1 TO LEN(SC$)
1210 CALL HCHAR(23,9+I,ASC(SEG$(SC$,I,1)))
1220 NEXT I
1230 GOTO 1250
1240 A=G
1250 IF (E(L,1)<>3)+(E(L,1)<>30)=-2 THEN 12
     90
1260 EY(L)=0
1270 EX(L)=SGN(X-E(L,0))
1280 GOTO 1300
1290 IF EG(L)=45 THEN 1260 ELSE 1320
1300 IF EX(L)<>0 THEN 1320
1310 EY(L)=SGN(Y-E(L,1))
1320 DEX=E(L,0)+EX(L)
1330 IF (DEX>=6)+(DEX<=20)=-2 THEN 1360
```

```
1340 EX(L)=0
1350 DEX=E(L,0)
1360 DEY=E(L,1)+EY(L)
1370 IF (DEY>=3)+(DEY<=30)=-2 THEN 1400
1380 EY(L)=0
1390 DEY=E(L,1)
1400 CALL GCHAR(DEX,DEY,EG1)
1410 IF EG1=112 THEN 1920
1420 IF EG1<>120 THEN 1510
1430 IF (E(L,2)<>4)+(E(L,2)<>28)=-2 THEN 14
     70
1440 EX(L)=2*SGN(X-E(L,1))
1450 EY(L)=0
1460 GOTO 1320
1470 EY(L)=2*SGN(Y-E(L,1))
1480 EX(L)=0
1490 IF EY(L)=0 THEN 860
1500 GOTO 1320
1510 IF EG1<>96 THEN 1600
1520 IF FL<>1 THEN 1560
1530 FL=0
1540 GOTO 860
1550 EG(L)=45
1560 EX(L)=0
1570 EY(L)=SGN(Y-E(L,1))
1580 FL=1
1590 GOTO 1320
1600 CALL HCHAR(E(L,0),E(L,1),EG(L))
1610 CALL HCHAR(DEX,DEY,120)
1620 E(L,0)=DEX
1630 E(L,1)=DEY
1640 EG(L)=EG1
1650 L=ABS(L-1)
1660 GOTO 860
1670 CALL SOUND(500,-8,2)
1680 IF SH=0 THEN 1250
1690 CALL SCREEN(7)
1700 CALL SCREEN(12)
1710 CALL SCREEN(14)
1720 CALL SCREEN(16)
1730 CALL SCREEN(4)
1740 FOR I=0 TO 1
1750 IF E(I,0)<>X THEN 1840
1760 CALL HCHAR(E(I,0),E(I,1),EG(I))
1770 E(I,0)=6
1780 E(I,1)=INT(RND*20)+4
1790 IF E(I,1)=E(ABS(I-1),1)THEN 1780
1800 EX(I)=0
1810 EY(I)=SGN(Y-E(I,1))
```

```
1820 CALL GCHAR(E(I,0),E(I,1),EG(I))
1830 CALL HCHAR(E(I,0),E(I,1),120)
1840 NEXT I
1850 SH=SH-1
1860 SH$=STR$(SH)&"  "
1870 FOR I=1 TO LEN(SH$)
1880 CALL HCHAR(23,24+I,ASC(SEG$(SH$,I,1)))
1890 NEXT I
1900 SC=SC-25
1910 GOTO 1190
1920 CALL SOUND(1000,-4,4)
1930 PRINT "N A B B E D !!";
1940 CALL HCHAR(X,Y-1,88,3)
1950 FOR D=1 TO 2000
1960 NEXT D
1970 PRINT ::"TRY AGAIN? (Y/N)";
1980 CALL KEY(0,K,S)
1990 IF K=89 THEN 370
2000 IF K<>78 THEN 1980
2010 CALL CLEAR
2020 END
```

4

Old Favorites

4

Match-Em

C. Regena

In addition to its primary purpose of captivating youngsters, this program also serves as a guide and example of how to create educational games on any subject.

This simple matching game is designed for young children. A screen of 16 squares is shown. Press the letters on two of the squares to try to match the shapes. If you "match-em," the shape will be drawn at the right side of the screen, and you won't be able to use those squares again (the shape is replaced by diagonal lines). There are eight pairs of shapes to try to match.

If you wish to stop the game at any time, press S and the placement of all the shapes will be shown. After each game you have the option of trying again—with the shapes scrambled in a different random order.

Other Applications

Take a look at the BASIC logic in this game, then design your own. You may wish to use the capabilities of the TI-99/4A graphics and draw other pictures—animals, people, designs, etc. Each shape here is drawn in a separate character set, and a random foreground and background color combination is chosen. Keep your drawing to eight or fewer graphics characters; you may also want to specify a certain foreground and background color.

You can make this matching game into an educational game. Instead of matching shapes, match an answer to a mathematics problem; match a capital city to its state; match a date to a historical event; match parts of a compound word. Whatever you want.

Programming Techniques

DIMensioned arrays start with a subscript of zero unless you specify OPTION BASE 1, which starts subscripts at 1. I used dimensioned numbers to keep track of the eight shapes (16 total) and various coordinates needed for graphics.

MX(n) and MY(n)—n is the number of the member of the array—are the X and Y coordinates to draw a shape at the right of the screen after it has been successfully matched. The coordinates

depend on how many matches have been made.

C1(n) and C2(n) are the X and Y coordinates for each square in the 16-square screen. D(n) indicates a red or a blue square.

A(n) and B(n) keep track of what shape is in which square. I use two arrays so that one can be a working array. B(n) also keeps track of the original order of the shapes when all the shapes are drawn (if you press S or if you have made all eight matches).

Lines 260-360 define graphics characters while the title screen is shown. Line 280 beeps a random sound for each character as it is defined. Graphics characters are defined by a string number. A null string is indicated either by "" or by two commas together and will yield a blank square for that graphics character. You do not need quote marks around the graphics string if it is in a DATA statement.

Lines 400-410 redefine the parentheses as a blue square and a red square. The game screen is then printed with lines 490-530. PRINTing characters is often faster than using the CALL HCHAR or CALL VCHAR method.

The shapes are numbered from 1 through 8. Lines 540-570 put the shape numbers in the B(n) array. Lines 590-650 mix up the numbers of the B(n) array and place them in the A(n) array. After a B(n) is chosen for the A(n) array, it is set to zero so it won't be chosen again. Lines 660-680 set the B(n) array equal to the A(n) array so that the shapes can all be printed in the original order at the end of the game.

Lines 700-750 choose a random foreground color and a random background color for each shape, making sure that the foreground color is not the same as the background color.

Lines 1480-1590 are a subroutine to draw the shape starting at coordinates X and Y. CH is the character number and is calculated in line 1480, depending on the shape number.

Explanation of the Program

Line Nos.	
110-120	DIMension variables starting with a subscript of 1.
130-160	Read X and Y coordinates for matched shapes.
170-210	Read X and Y coordinates and character numbers for each of the 16 squares.
220-230	Print title screen.
240-250	Define functions for random variables used later.
260-360	Define graphics characters for character numbers 96 through 159 (eight shapes, each in a different character set).

370-390	Print instructions.
400-410	Define characters for red and blue squares.
420-440	Wait for player to press any key.
450-470	Clear screen and initialize score (number of tries) and number of matches.
480-530	Define colors and draw game screen.
540-570	Define B(n) elements as shape numbers 1 through 8 (two of each number).
580	Print another line of game screen.
590-650	Randomly choose the order of the shapes in the 16 squares.
660-680	Set B(n) array elements equal to A(n) array.
690-750	Randomly choose colors for shapes.
760-780	Print name of game on screen.
790-830	Increment and print score.
840-890	Beep and wait for player to press a letter for first square.
900-980	Determine coordinates and draw diagonal lines if square has already been matched.
990	Call subroutine to draw shape.
1000-1070	Beep and wait for player to press a letter for second square.
1080-1150	Determine coordinates and draw diagonal lines if square has already been matched.
1160	Draw shape.
1170-1220	Determine if a match has been made; if not, sound "uh-oh."
1230-1300	If match has been made, these lines play arpeggio and determine coordinates, then draw shape at right of screen.
1310-1320	Set A(n) elements to zero so they cannot be used again for a correct match.
1330-1470	Cover squares again with red or blue square and return to next set of choices.
1480-1590	Subroutine to draw shape.
1600-1650	After all eight matches have been made, these lines play a tune.
1660-1710	Clear choices made and show all shapes on game screen.
1720-1750	Print option to play again, wait for player's choice, and branch appropriately.
1760-1770	Clear screen and end.

Match-Em

```
100 REM  MATCH-EM
110 OPTION BASE 1
120 DIM A(16),B(16),C1(16),C2(16),D(16),MX(
    8),MY(8)
```

4

```
130 FOR C=1 TO 8
140 READ MX(C),MY(C)
150 NEXT C
160 DATA 7,26,10,26,13,26,16,26,7,29,10,29,
    13,29,16,29
170 FOR C=1 TO 16
180 READ C1(C),C2(C),D(C)
190 NEXT C
200 DATA 3,5,40,3,10,41,3,15,40,3,20,41,8,5
    ,41,8,10,40,8,15,41,8,20,40
210 DATA 13,5,40,13,10,41,13,15,40,13,20,41
    ,18,5,41,18,10,40,18,15,41,18,20,40
220 CALL CLEAR
230 PRINT TAB(10);"MATCH-EM"::::::::::
240 DEF R=INT(RND*200+900)
250 DEF R15=INT(RND*15)+2
260 FOR C=96 TO 159
270 CALL SOUND(50,R,4)
280 READ C$
290 CALL CHAR(C,C$)
300 NEXT C
310 DATA "",,FFFFFFFFFFFFFFFF,,,,,,00000000
    00003CFF,0101030303030101,FFFFFFFFFFFFF
    FFF,8080C0C0C0C0808,,FF3C,,""
320 DATA 0000000008081C1C,0000000000000101,
    3E3E7F7FFFFFFFFF,000000008080C0C,030307
    07,FFFFFFFF,E0E0F0F,""
330 DATA "",0F0F0F0F0F0F0F0F,FFFFFFFFFFFFFF
    FF,F0F0F0F0F0F0F0F0,,,,,0000000010387CFE
    ,0103070F070301
340 DATA FFFFFFFFFFFFFFFE,0080C0E0C08,,7C38
    1,,,000000001010383C,001F070100010103,7
    CFFFFFFFEFFEFC7
350 DATA 00F0C0000000008,030706,8301,80C0C,
    ,0000000000003C7E,00010303030301,FFE7C3
    8181C3E7FF,0080C0C0C0C08,""
360 DATA 7E3C,,,000000003C3C3C3C,00000F0F0F
    0F,3C3CFFFFFFFF3C3C,0000F0F0F0F,,3C3C3C
    3C,,""
370 CALL CLEAR
380 PRINT "PRESS TWO LETTERS.":"TRY TO MAT
    CH THE SHAPES.":"THE BETTER YOU ARE, T
    HE"
390 PRINT :"LOWER YOUR SCORE WILL BE.":::"P
    RESS 'S' TO STOP THE GAME"::"AND SEE AL
    L THE SHAPES."
400 CALL CHAR(40,"FFFFFFFFFFFFFFFF")
410 CALL CHAR(41,"0")
420 PRINT :::"PRESS ANY KEY TO START.";
```

```
430 CALL KEY(0,K,S)
440 IF S<1 THEN 430
450 CALL CLEAR
460 SC=0
470 M=0
480 CALL COLOR(2,5,9)
490 PRINT "(((((((((((((((((((":"(((((((((
    ))))))))))":"((A(())B))((C(())D))":"((
    (((((((((((((((((("
500 PRINT "((((((((((((((((((":"))))))((((
    ()))))))(((((":")))))((((((((((((((":"))
    E))((F(())G))((H(("
510 PRINT "))))))((((((((((((((":")))))((((
    ()))))))(((((":"((((((((((((((((((":"((
    (((((((((((((((((("
520 PRINT "((I(())J))((K(())L))":"(((((((((
    ))))))))))":"((((((((((((((((((":"))
    )))))((((((((((((("
530 PRINT "))))))((((((((((((":"))M))((N(
    ())O))((P((":")))))((((((((((((":"))
    )))((((((((((("
540 FOR C=1 TO 8
550 B(C)=C
560 B(C+8)=C
570 NEXT C
580 PRINT :"S = STOP";TAB(20);"SCORE ="
590 FOR C=1 TO 16
600 RANDOMIZE
610 RC=INT(16*RND)+1
620 IF B(RC)=0 THEN 610
630 A(C)=B(RC)
640 B(RC)=0
650 NEXT C
660 FOR C=1 TO 16
670 B(C)=A(C)
680 NEXT C
690 M=0
700 FOR C=1 TO 8
710 F(C)=R15
720 F2(C)=R15
730 IF F2(C)=F(C)THEN 720
740 CALL COLOR(C+8,F(C),F2(C))
750 NEXT C
760 FOR C=1 TO 8
770 CALL HCHAR(2,23+C,ASC(SEG$("MATCH EM",C
    ,1)))
780 NEXT C
790 SC=SC+1
800 S$=STR$(SC)
```

```
810 FOR C=1 TO LEN(S$)
820 CALL HCHAR(23,27+C,ASC(SEG$(S$,C,1)))
830 NEXT C
840 CALL SOUND(150,1397,2)
850 CALL HCHAR(4,26,63)
860 CALL KEY(0,K,S)
870 IF K=83 THEN 1660
880 IF (K<65)+(K>80)THEN 860
890 CALL HCHAR(4,26,K)
900 N=K-64
910 A1=N
920 X=C1(N)
930 Y=C2(N)
940 IF A(N)<>0 THEN 990
950 CALL HCHAR(X,Y-1,92,3)
960 CALL HCHAR(X+1,Y-1,92,3)
970 CALL HCHAR(X+2,Y-1,92,3)
980 GOTO 1000
990 GOSUB 1480
1000 CALL SOUND(150,1397,2)
1010 CALL HCHAR(4,29,63)
1020 CALL KEY(0,K,S)
1030 IF K=83 THEN 1660
1040 IF (K<65)+(K>80)THEN 1020
1050 CALL HCHAR(4,29,K)
1060 N=K-64
1070 IF N=A1 THEN 1000
1080 A2=N
1090 X=C1(N)
1100 Y=C2(N)
1110 IF A(N)<>0 THEN 1160
1120 CALL HCHAR(X,Y-1,92,3)
1130 CALL HCHAR(X+1,Y-1,92,3)
1140 CALL HCHAR(X+2,Y-1,92,3)
1150 GOTO 1170
1160 GOSUB 1480
1170 IF A(A1)=0 THEN 1200
1180 IF A(A2)=0 THEN 1200
1190 IF A(A1)=A(A2)THEN 1230
1200 CALL SOUND(150,330,2)
1210 CALL SOUND(150,262,2)
1220 GOTO 1340
1230 M=M+1
1240 X=MX(M)
1250 Y=MY(M)
1260 CALL SOUND(150,262,2)
1270 CALL SOUND(150,330,2)
1280 CALL SOUND(150,392,2)
1290 CALL SOUND(300,523,2)
```

```
1300 GOSUB 1500
1310 A(A1)=0
1320 A(A2)=0
1330 IF M=8 THEN 1600
1340 X=C1(A2)
1350 Y=C2(A2)
1360 CALL HCHAR(X,Y-1,D(N),3)
1370 CALL HCHAR(X+1,Y-1,D(N),3)
1380 CALL HCHAR(X+2,Y-1,D(N),3)
1390 CALL HCHAR(X+1,Y,N+64)
1400 X=C1(A1)
1410 Y=C2(A1)
1420 CALL HCHAR(X,Y-1,D(A1),3)
1430 CALL HCHAR(X+1,Y-1,D(A1),3)
1440 CALL HCHAR(X+2,Y-1,D(A1),3)
1450 CALL HCHAR(X+1,Y,A1+64)
1460 CALL HCHAR(4,26,32,4)
1470 GOTO 790
1480 CH=8*(B(N)-1)+96
1490 CALL SOUND(150,-1,2)
1500 CALL HCHAR(X,Y-1,CH+7)
1510 CALL HCHAR(X,Y,CH)
1520 CALL HCHAR(X,Y+1,CH+7)
1530 CALL HCHAR(X+1,Y-1,CH+1)
1540 CALL HCHAR(X+1,Y,CH+2)
1550 CALL HCHAR(X+1,Y+1,CH+3)
1560 CALL HCHAR(X+2,Y-1,CH+4)
1570 CALL HCHAR(X+2,Y,CH+5)
1580 CALL HCHAR(X+2,Y+1,CH+6)
1590 RETURN
1600 RESTORE 1610
1610 DATA 262,330,392,523,330,392,523,659,3
     92,523,659,784,523,659,784,1046,1046
1620 FOR C=1 TO 17
1630 READ J
1640 CALL SOUND(-99,J,2)
1650 NEXT C
1660 CALL HCHAR(4,26,32,4)
1670 FOR N=1 TO 16
1680 X=C1(N)
1690 Y=C2(N)
1700 GOSUB 1480
1710 NEXT N
1720 PRINT :"PLAY AGAIN? [Y N]";
1730 CALL KEY(0,K,S)
1740 IF K=78 THEN 1760
1750 IF K=89 THEN 450 ELSE 1730
1760 CALL CLEAR
1770 END
```

4

Tic-Tac-Toe

C. Regena

This classic game pits player against computer. On the professional level the computer can be a very formidable opponent.

Playing Tic-Tac-Toe

First choose whether you wish to play at an amateur level or at a professional level. Next choose either the X markers or the O markers. The X markers have the first move, and you take turns with the computer. Try to get three markers in a row—across, down, or diagonally—and at the same time prevent the computer from putting three markers in a row.

Program Explanation

Lines 150-220 define the graphics and colors used in the game. Lines 230-260 read in the X and Y coordinates of each of the nine positions in the Tic-Tac-Toe grid. Lines 270-290 initialize the values of each of the nine positions to zero. Later, all your markers will have a value of 1, and all computer markers will have a value of −1. The color of Tic-Tac-Toe bars is randomly chosen for each game at lines 480-530. Lines 540-560 number the squares, so you can press the number of the position you want for your turn. The O markers are drawn with lines 1660-1700, and the X markers are drawn with lines 1720-1800.

For the amateur level, the computer just randomly chooses one of the unoccupied positions on the grid (lines 2180-2220). On the professional level, the computer takes into account where markers are already placed and chooses a position. The computer's strategy is in lines 590-1520. Lines 1530-1600 receive your desired marker position.

Lines 1810-1860 check to see if there are three of the same type of marker in any row. Lines 1870-1920 check for three of the same type of marker in any column. Lines 1930-2000 check for a diagonal win or a tie game.

Tic-Tac-Toe

```
100 REM  TIC-TAC-TOE
110 REM
```

```
120 CALL CLEAR
130 PRINT TAB(9);"TIC-TAC-TOE"
140 PRINT :::"TRY TO GET THREE IN A ROW.":::
150 FOR C=104 TO 114
160 READ C$
170 CALL CHAR(C,C$)
180 NEXT C
190 DATA 00030C102020404,FF,00C0300804040202
    ,808080808080808,0101010101010101,404020
    20100C03
200 DATA 00000000000000FF,020204040830C,8040
    201008040201,010204081020408,81422418182
    44281
210 CALL COLOR(10,7,1)
220 CALL COLOR(11,5,1)
230 FOR C=1 TO 9
240 READ X(C),Y(C)
250 NEXT C
260 DATA 6,9,6,15,6,21,12,9,12,15,12,21,18,9
    ,18,15,18,21
270 FOR I=1 TO 9
280 T(I)=0
290 NEXT I
300 FL=0
310 PRINT :"PRESS 1 FOR AMATEUR"
320 PRINT TAB(7);"2 FOR PROFESSIONAL"
330 CALL KEY(0,K,S)
340 IF (K<49)+(K>50)THEN 330
350 L=K-48
360 PRINT ::"PRESS 1 -- X, FIRST MOVE"
370 PRINT TAB(7);"2 -- 0, SECOND MOVE"
380 CALL SOUND(150,1497,2)
390 CALL KEY(0,K,S)
400 IF (K<49)+(K>50)THEN 390
410 C=K-48
420 CALL CLEAR
430 FOR I=1 TO 11
440 RANDOMIZE
450 CALL SOUND(-100,INT(RND*900+500),2)
460 CALL HCHAR(2,10+I,ASC(SEG$("TIC-TAC-TOE"
    ,I,1)))
470 NEXT I
480 CO=INT(RND*8)+9
490 CALL COLOR(9,CO,CO)
500 CALL VCHAR(5,13,96,17)
510 CALL VCHAR(5,19,96,17)
520 CALL HCHAR(10,8,96,17)
530 CALL HCHAR(16,8,96,17)
540 FOR I=1 TO 9
```

```
550 CALL HCHAR(X(I)+1,Y(I)+1,48+I)
560 NEXT I
570 IF L=1 THEN 2180
580 IF C=1 THEN 1530
590 A=-1
600 B=1
610 IF T(5)<>0 THEN 650
620 T(5)=-1
630 M=5
640 GOTO 1510
650 IF T(5)<>1 THEN 700
660 IF T(1)<>0 THEN 740
670 T(1)=-1
680 M=1
690 GOTO 1510
700 IF (T(2)=1)+(T(1)=0)=-2 THEN 1390
710 IF (T(4)=1)+(T(1)=0)=-2 THEN 1390
720 IF (T(6)=1)+(T(9)=0)=-2 THEN 1490
730 IF (T(8)=1)+(T(9)=0)=-2 THEN 1490
740 IF A<>1 THEN 830
750 D=3*INT((M-1)/3)+1
760 IF D<>M THEN 780
770 L=1
780 IF D+1<>M THEN 800
790 L=2
800 IF D+2<>M THEN 850
810 L=3
820 GOTO 850
830 FOR D=1 TO 7 STEP 3
840 FOR L=1 TO 3
850 IF T(D)<>A THEN 910
860 IF T(D+2)<>A THEN 970
870 IF T(D+1)<>0 THEN 1020
880 T(D+1)=-1
890 M=D+1
900 GOTO 1510
910 IF T(D)=B THEN 1020
920 IF T(D+2)<>A THEN 1020
930 IF T(D+1)<>A THEN 1020
940 T(D)=-1
950 M=D
960 GOTO 1510
970 IF T(D+2)<>0 THEN 1020
980 IF T(D+1)<>A THEN 1020
990 T(D+2)=-1
1000 M=D+2
1010 GOTO 1510
1020 IF T(L)<>A THEN 1080
1030 IF T(L+6)<>A THEN 1140
```

```
1040 IF T(L+3)<>0 THEN 1190
1050 T(L+3)=-1
1060 M=L+3
1070 GOTO 1510
1080 IF T(L)=B THEN 1190
1090 IF T(L+6)<>A THEN 1190
1100 IF T(L+3)<>A THEN 1190
1110 T(L)=-1
1120 M=L
1130 GOTO 1510
1140 IF T(L+6)<>0 THEN 1190
1150 IF T(L+3)<>A THEN 1190
1160 T(L+6)=-1
1170 M=L+6
1180 GOTO 1510
1190 IF A=1 THEN 1230
1200 IF (D=7)+(L=3)=-2 THEN 1230
1210 NEXT L
1220 NEXT D
1230 IF T(5)<>A THEN 1280
1240 IF (T(3)=A)+(T(7)=0)=-2 THEN 1460
1250 IF (T(9)=A)+(T(1)=0)=-2 THEN 1390
1260 IF (T(7)=A)+(T(3)=0)=-2 THEN 1430
1270 IF (T(9)=0)+(T(1)=A)=-2 THEN 1490
1280 IF A<>-1 THEN 1320
1290 A=1
1300 B=-1
1310 GOTO 740
1320 IF (T(9)=1)+(T(3)=0)=-2 THEN 1420
1330 FOR I=2 TO 9
1340 IF T(I)<>0 THEN 1380
1350 T(I)=-1
1360 M=I
1370 GOTO 1510
1380 NEXT I
1390 T(1)=-1
1400 M=1
1410 GOTO 1510
1420 IF T(1)=1 THEN 1330
1430 T(3)=-1
1440 M=3
1450 GOTO 1510
1460 T(7)=-1
1470 M=7
1480 GOTO 1510
1490 T(9)=-1
1500 M=9
1510 CALL SOUND(100,440,2)
1520 GOSUB 1640
```

```
1530 CALL SOUND(100,1497,2)
1540 CALL KEY(0,K,S)
1550 IF S<1 THEN 1540
1560 IF (K<49)+(K>57)THEN 1530
1570 M=K-48
1580 IF T(M)<>0 THEN 1530
1590 A=1
1600 T(M)=1
1610 GOSUB 1640
1620 IF L=1 THEN 2190 ELSE 590
1630 STOP
1640 C=ABS(C-1)
1650 ON C+1 GOTO 1660,1720
1660 CALL HCHAR(X(M),Y(M),112)
1670 CALL HCHAR(X(M),Y(M)+2,113)
1680 CALL HCHAR(X(M)+1,Y(M)+1,114)
1690 CALL HCHAR(X(M)+2,Y(M),113)
1700 CALL HCHAR(X(M)+2,Y(M)+2,112)
1710 GOTO 1810
1720 CALL HCHAR(X(M),Y(M),104)
1730 CALL HCHAR(X(M),Y(M)+1,105)
1740 CALL HCHAR(X(M),Y(M)+2,106)
1750 CALL HCHAR(X(M)+1,Y(M),107)
1760 CALL HCHAR(X(M)+1,Y(M)+2,108)
1770 CALL HCHAR(X(M)+2,Y(M),109)
1780 CALL HCHAR(X(M)+2,Y(M)+1,110)
1790 CALL HCHAR(X(M)+2,Y(M)+2,111)
1800 CALL HCHAR(X(M)+1,Y(M)+1,32)
1810 FOR I=1 TO 7 STEP 3
1820 IF T(I)<>T(I+1)THEN 1860
1830 IF T(I)<>T(I+2)THEN 1860
1840 IF T(I)=-1 THEN 2050
1850 IF T(I)=1 THEN 2030
1860 NEXT I
1870 FOR I=1 TO 3
1880 IF T(I)<>T(I+3)THEN 1920
1890 IF T(I)<>T(I+6)THEN 1920
1900 IF T(I)=-1 THEN 2050
1910 IF T(I)=1 THEN 2030
1920 NEXT I
1930 FOR I=1 TO 9
1940 IF T(I)=0 THEN 1970
1950 NEXT I
1960 FL=1
1970 IF T(5)<>A THEN 2000
1980 IF (T(1)=A)+(T(9)=A)=-2 THEN 2020
1990 IF (T(3)=A)+(T(7)=A)=-2 THEN 2020
2000 IF FL=1 THEN 2070
2010 RETURN
```

```
2020 IF A=-1 THEN 2050
2030 PRINT "YOU WON!!"
2040 GOTO 2080
2050 PRINT "COMPUTER WINS!!"
2060 GOTO 2080
2070 PRINT "IT'S A TIE."
2080 FOR I=1 TO 20
2090 CALL SOUND(100,INT(RND*500)+500,2)
2100 NEXT I
2110 PRINT :"TRY AGAIN? (Y/N)"
2120 CALL SOUND(100,1497,2)
2130 CALL KEY(0,K,S)
2140 IF K=89 THEN 270
2150 IF K<>78 THEN 2130
2160 CALL CLEAR
2170 STOP
2180 IF C=1 THEN 1530
2190 M=INT(RND*9)+1
2200 IF T(M)<>0 THEN 2190
2210 A=-1
2220 T(M)=-1
2230 GOSUB 1640
2240 GOTO 1530
2250 END
```

4

Boggler

Gary Braun
Translated for the TI by Charles Brannon

In the tradition of popular board games like checkers, "Boggler" should offer hours of intriguing strategic planning as you try to capture your opponent's pieces and avoid capture yourself.

This is a simple, yet challenging, game of skill. The object of the game is to capture five pairs of your opponent's pebbles or get five consecutive pebbles in a row, either vertically, horizontally, or diagonally. In this version, the computer acts as the playing board, checking all moves for validity and keeping score for the two opposing players, who alternate turns.

At the bottom of the screen is a line that turns either red or blue to indicate whose turn it is. The computer drops the first pebble for red by placing it at the center of the board. The arrow keys are used to position the player's pebble in the desired location, and the ENTER key is used to drop the pebble.

The program then checks to see if the player has captured a pair of his opponent's pebbles by placing one of his on either side, or if he has five pebbles in a row. If the player has captured two pebbles, the program places them on his side of the board.

Two final notes on the program: it is not possible to move off the game board; and, as you take a new turn, the program will always return you to where you made *your* last play.

Boggler

```
100 REM BOGGLER
110 REM
120 DIM DELTAX(8),DELTAY(8),ROW(1),COL(1),PI
    ECE(1),COUNT(1)
130 GOSUB 980
140 REM MAIN LOOP
150 PLR=1-PLR
160 CALL HCHAR(24,1,128+PLR*8,32)
170 CALL GCHAR(ROW(PLR),COL(PLR),PK)
180 CALL HCHAR(ROW(PLR),COL(PLR),PIECE(PLR))
190 REM
200 CALL KEY(3,WHICH,STATUS)
```

```
210 IF (WHICH=13)*(PK=144)THEN 340
220 CALL HCHAR(ROW(PLR),COL(PLR),32)
230 IF STATUS=0 THEN 180
240 CALL HCHAR(ROW(PLR),COL(PLR),PK)
250 TEMPROW=ROW(PLR)
260 TEMPCOL=COL(PLR)
270 ROW(PLR)=ROW(PLR)+(WHICH=69)-(WHICH=88)
280 COL(PLR)=COL(PLR)+(WHICH=83)-(WHICH=68)
290 CALL GCHAR(ROW(PLR),COL(PLR),PK)
300 IF PK<>152 THEN 170
310 ROW(PLR)=TEMPROW
320 COL(PLR)=TEMPCOL
330 GOTO 170
340 CALL HCHAR(ROW(PLR),COL(PLR),PIECE(PLR))
350 REM CHECK FOR FIVE-IN-A-ROW
360 REM
370 REM STEP 1. FIND BOTTOM
380 FOR I=1 TO 8
390 TEMPROW=ROW(PLR)
400 TEMPCOL=COL(PLR)
410 DX=DELTAX(I)
420 DY=DELTAY(I)
430 N=0
440 TEMPCOL=TEMPCOL+DX
450 TEMPROW=TEMPROW+DY
460 CALL GCHAR(TEMPROW,TEMPCOL,PK)
470 IF PK<>PIECE(PLR)THEN 500
480 N=N+1
490 GOTO 440
500 N=0
510 TEMPCOL=TEMPCOL-DX
520 TEMPROW=TEMPROW-DY
530 CALL GCHAR(TEMPROW,TEMPCOL,PK)
540 IF PK<>PIECE(PLR)THEN 570
550 N=N+1
560 GOTO 510
570 IF N=5 THEN 800
580 NEXT I
590 REM NO FIVE-IN-A-ROW
600 REM CHECK FOR CAPTURE
610 FOR I=1 TO 8
620 TEMPROW=ROW(PLR)
630 TEMPCOL=COL(PLR)
640 DX=DELTAX(I)
650 DY=DELTAY(I)
660 CALL GCHAR(TEMPROW+DY,TEMPCOL+DX,CK1)
670 CALL GCHAR(TEMPROW+DY*2,TEMPCOL+DX*2,CK2
    )
680 CALL GCHAR(TEMPROW+DY*3,TEMPCOL+DX*3,CK3)
```

```
690 IF (CK1<>PIECE(1-PLR))+(CK2<>PIECE(1-PLR
    ))+(CK3<>PIECE(PLR))THEN 770
700 COUNT(PLR)=COUNT(PLR)+2
710 IF COUNT(PLR)>10 THEN 720 ELSE 730
720 COUNT(PLR)=10
730 CALL VCHAR(5,28*PLR+2,PIECE(1-PLR),COUNT
    (PLR))
740 CALL HCHAR(TEMPROW+DY,TEMPCOL+DX,144)
750 CALL HCHAR(TEMPROW+DY*2,TEMPCOL+DX*2,144
    )
760 IF COUNT(PLR)=10 THEN 800
770 NEXT I
780 GOTO 150
790 REM SOMEBODY WON!
800 FOR I=1 TO 20
810 CALL SCREEN(RND*15+1)
820 NEXT I
830 FOR I=1 TO 10
840 CALL COLOR(14-PLR,RND*15+1,RND*15+1)
850 CALL SOUND(-1,I*100+100,I*3)
860 NEXT I
870 CALL COLOR(14-PLR,1,1)
880 CALL CLEAR
890 PRINT "PLAYER ";
900 IF PLR THEN 930
910 PRINT "TWO";
920 GOTO 940
930 PRINT "ONE";
940 PRINT " WINS!"
950 END
960 REM INITIALIZATION
970 REM
980 CALL SCREEN(16)
990 FOR I=1 TO 8
1000 READ A,B
1010 DELTAX(I)=A
1020 DELTAY(I)=B
1030 NEXT I
1040 DATA 0,1,1,0,1,1,-1,-1,-1,0,0,-1,-1,1,1
     ,-1
1050 CALL CLEAR
1060 CALL COLOR(13,9,1)
1070 CALL COLOR(14,6,1)
1080 CALL COLOR(15,14,1)
1090 CALL COLOR(16,12,1)
1100 CALL CHAR(152,"FEFEFEFEFEFEFE00")
1110 CALL CHAR(144,"007E666666667E00")
1120 CALL CHAR(129,"007E7E7E7E7E7E00")
1130 CALL CHAR(137,"007E7E7E7E7E7E00")
```

```
1140 CALL CHAR(128,"FFFFFFFFFFFFFFFF")
1150 CALL CHAR(136,"FFFFFFFFFFFFFFFF")
1160 CALL HCHAR(3,3,152,26)
1170 CALL HCHAR(22,3,152,26)
1180 CALL VCHAR(3,3,152,20)
1190 CALL VCHAR(3,29,152,20)
1200 FOR I=4 TO 21
1210 CALL HCHAR(I,4,144,25)
1220 NEXT I
1230 PIECE(0)=129
1240 PIECE(1)=137
1250 COUNT(0)=0
1260 COUNT(1)=0
1270 ROW(0)=12
1280 COL(0)=16
1290 ROW(1)=11
1300 COL(1)=16
1310 CALL HCHAR(ROW(0),COL(0),PIECE(0))
1320 RETURN
```

4

Flip Flop

C. Regena

Planning and thought make "Flip Flop" a game that people have enjoyed for years.

"Flip Flop" is a board game is which you alternate plays with the computer. To make a legal move, you must sandwich in at least one square of the opposite color between your new move and any previous square of your color. You thus capture any colored squares between your two squares, and the captured squares are changed to your color—the colors are reversed. You may capture horizontally, vertically, or diagonally, or in several directions at once. If a legal move cannot be made, it is the other player's turn.

Use the arrow keys to move the cursor to your desired position, then press ENTER. If it is a legal move, your marker will replace the cursor, and the appropriate squares will be reversed.

The game is over when the board is filled, when one player has all the squares on the board, or when it is impossible for either player to move. The player with the most squares is the winner.

Program Explanation

The computer's strategy in this version of the board game is simply to capture the maximum squares in a particular move—without looking ahead. You may prefer to consider logic that checks edge and corner positions. You can place characters of different numbers in those positions, or perhaps place blank characters all around the edges then adjust the N value to reflect advantageous conditions.

Lines 140-200 define the graphics characters for the board and the markers. Lines 210-230 define directions which are used to determine a legal move. Lines 240-250 define X and Y as the coordinates on the board.

Lines 330-410 draw the starting board on the screen with the starting four markers in the standard positions. Lines 350-370 mark surrounding squares which are possible moves. T is the number of turns, or the number of markers on the board. P is 1 or 2 for first player or second player.

Lines 440-680 contain the procedure for moving the cursor

with the arrow keys for your move. Line 690 makes sure your move is to a possible square (character 96). If the move was not legal, lines 770-850 see if there is a possible legal move. If there is no legal move, then switch players. Lines 880-940 mark the new surrounding squares with character 96 because they become possible squares for legal moves.

Lines 960-1050 check for game-ending conditions—either a filled board or all squares of the same color.

Lines 1060-1080 change the player number and the corresponding graphics character number and then branch to either the player's move or the computer's move.

Lines 1090-1290 contain the procedure for the computer's move. Z is the largest number of squares captured, and N is the number of squares captured for a particular move.

Lines 1300-1410 contain a subroutine to check whether there is a legal move. Lines 1420-1730 are a subroutine to count the number of squares captured and to keep track of the coordinates of captured squares. Lines 1740-1770 are a subroutine to reverse the colored squares captured.

Lines 1780-2060 count the number of squares of each color then display an ending message.

Flip Flop

```
110 CALL CLEAR
120 PRINT TAB(7);"** FLIP FLOP **"::::"USE T
    HE ARROW KEYS TO MOVE"::"THEN PRESS <ENT
    ER>.":: 
130 DIM D(3),NX(18),NY(18)
140 FOR C=96 TO 128 STEP 8
150 CALL CHAR(C,"FF818181818181FF")
160 NEXT C
170 CALL COLOR(11,2,16)
180 CALL COLOR(12,2,6)
190 CALL COLOR(13,2,7)
200 C$="{10 SPACES}hhhhhhhh"
210 D(1)=-1
220 D(2)=0
230 D(3)=1
240 X=11
250 Y=17
260 PRINT ::"{3 SPACES}p YOUR MARKERS"::"
    {3 SPACES}x COMPUTER"
270 PRINT ::"CHOOSE:"::"{3 SPACES}1  FIRST M
    OVE"::"{3 SPACES}2  SECOND MOVE"
280 CALL KEY(0,K,S)
```

```
290 IF (K<49)+(K>50)THEN 280
300 P=K-48
310 P1=P
320 G1=104+P*8
330 CALL CLEAR
340 PRINT C$:C$:C$:C$:C$:C$:C$:C$::::::::
350 FOR C=11 TO 14
360 CALL HCHAR(C,15,96,4)
370 NEXT C
380 CALL HCHAR(12,16,112)
390 CALL HCHAR(12,17,120)
400 CALL HCHAR(13,16,120)
410 CALL HCHAR(13,17,112)
420 T=5
430 IF P=2 THEN 1100
440 CALL GCHAR(X,Y,GA)
450 CALL KEY(0,K,S)
460 CALL HCHAR(X,Y,128)
470 CALL HCHAR(X,Y,GA)
480 IF K=13 THEN 690
490 IF K<>69 THEN 520
500 X=X-1
510 GOTO 600
520 IF K<>68 THEN 550
530 Y=Y+1
540 GOTO 600
550 IF K<>83 THEN 580
560 Y=Y-1
570 GOTO 600
580 IF K<>88 THEN 450
590 X=X+1
600 IF X>8 THEN 620
610 X=9
620 IF X<17 THEN 640
630 X=16
640 IF Y>12 THEN 660
650 Y=13
660 IF Y<21 THEN 680
670 Y=20
680 GOTO 440
690 IF GA=96 THEN 730
700 CALL SOUND(50,330,2)
710 CALL SOUND(50,262,2)
720 GOTO 450
730 CALL SOUND(100,1400,2)
740 CALL HCHAR(X,Y,128)
750 GOSUB 1420
760 IF N>0 THEN 860
770 GOSUB 1300
```

```
780 IF FL=0 THEN 700
790 CALL SOUND(500,-1,4)
800 FOR C=1 TO 7
810 CALL HCHAR(24,C+2,ASC(SEG$("NO MOVE",C,1
    )))
820 NEXT C
830 CALL SOUND(1,-1,30)
840 CALL HCHAR(24,3,32,7)
850 GOTO 1060
860 CALL HCHAR(X,Y,G1)
870 CALL SOUND(100,523,4)
880 FOR I=X-1 TO X+1
890 FOR J=Y-1 TO Y+1
900 CALL GCHAR(I,J,G)
910 IF G<>104 THEN 930
920 CALL HCHAR(I,J,96)
930 NEXT J
940 NEXT I
950 GOSUB 1740
960 T=T+1
970 IF T=65 THEN 1780
980 FOR I=9 TO 16
990 FOR J=13 TO 20
1000 CALL GCHAR(I,J,G)
1010 IF G<=104 THEN 1030
1020 IF G<>G1 THEN 1060
1030 NEXT J
1040 NEXT I
1050 GOTO 1780
1060 P=1-SGN(P-2)
1070 G1=P*8+104
1080 ON P GOTO 440,1100
1090 REM  COMPUTER
1100 Z=-1
1110 FOR X=9 TO 16
1120 FOR Y=13 TO 20
1130 X1=X-8
1140 Y1=Y-12
1150 CALL GCHAR(X,Y,G)
1160 IF G<>96 THEN 1230
1170 GOSUB 1420
1180 IF N=0 THEN 1230
1190 IF N<=Z THEN 1230
1200 Z=N
1210 IX=X
1220 IY=Y
1230 NEXT Y
1240 NEXT X
1250 IF Z<=0 THEN 790
```

```
1260 X=IX
1270 Y=IY
1280 CALL HCHAR(X,Y,G1)
1290 GOTO 750
1300 FL=0
1310 FOR II=9 TO 16
1320 FOR JJ=13 TO 20
1330 CALL GCHAR(II,JJ,G)
1340 IF G<>96 THEN 1390
1350 GOSUB 1420
1360 IF N=0 THEN 1390
1370 FL=1
1380 GOTO 1410
1390 NEXT JJ
1400 NEXT II
1410 RETURN
1420 N=1
1430 FOR C=1 TO 3
1440 FOR A=1 TO 3
1450 I=X+D(C)
1460 J=Y+D(A)
1470 IF I<>X THEN 1490
1480 IF J=Y THEN 1700
1490 CALL GCHAR(I,J,G)
1500 IF P=1 THEN 1520
1510 IF G=112 THEN 1530 ELSE 1700
1520 IF G<>120 THEN 1700
1530 I=I+D(C)
1540 J=J+D(A)
1550 CALL GCHAR(I,J,G)
1560 IF G<=104 THEN 1700
1570 IF P=1 THEN 1590
1580 IF G=120 THEN 1600 ELSE 1530
1590 IF G<>112 THEN 1530
1600 I=X+D(C)
1610 J=Y+D(A)
1620 CALL GCHAR(I,J,G)
1630 IF G=G1 THEN 1700
1640 NX(N)=I
1650 NY(N)=J
1660 N=N+1
1670 I=I+D(C)
1680 J=J+D(A)
1690 GOTO 1620
1700 NEXT A
1710 NEXT C
1720 N=N-1
1730 RETURN
1740 FOR C=1 TO N
```

```
1750 CALL HCHAR(NX(C),NY(C),G1)
1760 NEXT C
1770 RETURN
1780 T1=0
1790 T2=0
1800 FOR I=9 TO 16
1810 FOR J=13 TO 20
1820 CALL GCHAR(I,J,G)
1830 IF G<112 THEN 1900
1840 IF G=120 THEN 1880
1850 T1=T1+1
1860 CALL SOUND(-50,523,4)
1870 GOTO 1900
1880 T2=T2+1
1890 CALL SOUND(-50,262,4)
1900 NEXT J
1910 NEXT I
1920 PRINT CHR$(112);T1
1930 PRINT CHR$(120);T2
1940 IF T1<>T2 THEN 1970
1950 PRINT :"TIE GAME"
1960 GOTO 2010
1970 IF T1>T2 THEN 2000
1980 PRINT :CHR$(120);" COMPUTER WINS"
1990 GOTO 2010
2000 PRINT :CHR$(112);" YOU WIN"
2010 PRINT :"PLAY AGAIN? (Y/N)";
2020 CALL KEY(0,K,S)
2030 IF K=89 THEN 240
2040 IF K<>78 THEN 2020
2050 CALL CLEAR
2060 END
```

5

Thinking Games

5

Color Codes

C. Regena

Will you break the computer's code? Can you find the secret sequence? The computer will give you nine tries and even a few clues to help.

The computer will choose a series of four colors. Each position may be one of six colors. You try to guess the computer's color codes by pressing the color numbers:

1	White	4	Blue
2	Green	5	Purple
3	Yellow	6	Red

If you make a mistake before you press the fourth color number, press 0 (zero) to erase your entire guess and start over.

The computer will check your guess. You will get one black marker for each correct color that is also in the correct position. You will get a white marker for a color that is in the series but not in the correct position. The black and white markers are not in any particular order—you may not know *which* color goes with which marker. Try to guess the color codes in as few tries as possible. After your nine guesses, the computer will show you the correct answer.

Program Explanation

Lines 230-340 define the solid squares of color. Lines 420-450 choose the computer's series of colors. Lines 460-520 print the color numbers at the top of the screen. Lines 580-730 receive your four color choices. Lines 750-820 check to see if any of the colors you guessed are the same color and in the same position as in the computer's sequence. If so, a black marker is printed, and that color guess will not be checked later. Lines 840-930 check to see if any colors you guessed are correct colors (but in the wrong position) for the white markers.

Color Codes

```
100 REM  COLOR CODES
110 REM  BY REGENA
120 CALL CLEAR
```

```
130 PRINT TAB(7);"COLOR CODES"
140 CALL CHAR(93,"3C7EFFFFFFFF7E3C")
150 PRINT ::::"GUESS THE CORRECT SEQUENCE"
160 PRINT "OF FOUR COLORS."
170 CALL CHAR(120,"3C7EFFFFFFFF7E3C")
180 PRINT :"GUESS BY PRESSING THE"
190 PRINT "COLOR NUMBERS."
200 CALL COLOR(12,16,1)
210 PRINT :::CHR$(93);" RIGHT COLOR AND POSITION"
220 PRINT :CHR$(120);" RIGHT COLOR ONLY"
230 FOR I=1 TO 3
240 CALL CHAR((11+I)*8,"0")
250 CALL CHAR((11+I)*8+1,"FFFFFFFFFFFFFFFF")
260 NEXT I
270 FOR I=1 TO 6
280 READ C(I)
290 NEXT I
300 DATA 96,97,104,105,112,113
310 CALL COLOR(9,3,16)
320 CALL COLOR(10,5,12)
330 CALL COLOR(11,7,14)
340 CALL CHAR(92,"FFFFFFFFFFFFFFFF")
350 PRINT :::"PRESS ANY KEY TO START.";
360 CALL KEY(0,K,S)
370 IF S<1 THEN 360
380 CALL CLEAR
390 FOR I=9 TO 17
400 CALL VCHAR(4,I,92,20)
410 NEXT I
420 FOR I=1 TO 4
430 RANDOMIZE
440 A(I)=INT(RND*6+1)
450 NEXT I
460 FOR I=1 TO 6
470 CALL HCHAR(1,3*I-1,48+I)
480 CALL HCHAR(1,3*I,C(I))
490 NEXT I
500 FOR I=1 TO 7
510 CALL HCHAR(1,21+I,ASC(SEG$("0-ERASE",I,1)))
520 NEXT I
530 I=0
540 I=I+1
550 IF I>9 THEN 950
560 IA=3+2*I
570 CALL HCHAR(IA,8,48+I)
580 FOR J=1 TO 4
590 JA=8+2*J
600 CALL SOUND(100,1497,2)
610 CALL KEY(0,K,S)
620 CALL HCHAR(IA,JA,92)
```

```
630 CALL HCHAR(IA,JA,32)
640 IF S<1 THEN 610
650 IF (K>47)*(K<55)<>1 THEN 610
660 IF K<>48 THEN 710
670 FOR K=J TO 1 STEP -1
680 CALL HCHAR(IA,8+K*2,92)
690 NEXT K
700 GOTO 580
710 B(J)=K-48
720 CALL HCHAR(IA,JA,C(K-48))
730 NEXT J
740 L=1
750 FOR J=1 TO 4
760 D(J)=A(J)
770 IF A(J)<>B(J)THEN 820
780 CALL HCHAR(IA,21+L,93)
790 B(J)=0
800 D(J)=8
810 L=L+1
820 NEXT J
830 IF L=5 THEN 1010
840 FOR J=1 TO 4
850 FOR J1=1 TO 4
860 IF D(J)<>B(J1)THEN 920
870 CALL HCHAR(IA,21+L,120)
880 B(J1)=0
890 D(J)=8
900 L=L+1
910 IF L=5 THEN 540
920 NEXT J1
930 NEXT J
940 GOTO 540
950 CALL SOUND(150,330,2)
960 CALL SOUND(150,262,2)
970 FOR J=1 TO 4
980 CALL HCHAR(23,8+2*J,C(A(J)))
990 NEXT J
1000 GOTO 1060
1010 FOR J=1 TO 3
1020 CALL SOUND(100,262,2)
1030 CALL SOUND(100,330,2)
1040 CALL SOUND(100,392,2)
1050 NEXT J
1060 PRINT "TRY AGAIN? (Y/N)";
1070 CALL KEY(0,K,S)
1080 IF K=89 THEN 380
1090 IF K<>78 THEN 1070
1100 CALL CLEAR
1110 END
```

5

Copycat

Mark and Dan Powell
Translated for the TI by C. Regena

"Copycat" is an entertaining, musical, and colorful "match-me" game. It exercises pattern recognition, short-term memory and hand-eye coordination, making it an excellent game for all ages.

"Copycat" is a memory game. The computer plays one of four tones, and you match the tone by pressing the correct number, 1, 2, 3, or 4. The computer then plays two tones. You match the series by pressing two numbers in the correct order. Each round adds one more random tone, and you need to remember the series to copy. How long can you be a copycat?

Program Explanation

The colors for the four squares are defined in lines 260-290, but the colors can be in a different order for each game. The colors corresponding to the numbered squares are chosen in lines 520-580. The four tones, T, are defined in lines 300-330.

The word MISS is printed on the screen invisibly in lines 490-510. When a miss occurs, the color set containing the characters 97-99 blinks red, yellow, and black (lines 910-950), then the word is turned invisible again (line 960). M is the number of misses; if M is equal to 3, the game ends.

L(n) is the array that holds the tone numbers or square numbers as they are chosen randomly. Line 640 chooses a random number from 1 to 4 for the next tone in the series. Lines 650-710 play each tone and light up each square in the series. Lines 720-810 detect which number is pressed and compare that number with the correct number in the series.

Lines 1000-1060 present the option to play the game again or to end the program.

Copycat

```
100 REM  COPYCAT
110 CALL CLEAR
120 PRINT TAB(8);"C O P Y C A T"
130 DIM L(100)
```

```
140 DEF R4=INT(RND*4+1)
150 PRINT :::::"{3 SPACES}PRESS 1-4 TO COPY
    THE"
160 PRINT :"{3 SPACES}COMPUTER."
170 PRINT :::"{3 SPACES}THE GAME ENDS AFTER"
180 PRINT :"{3 SPACES}THREE MISSES.":::::
190 FOR I=9 TO 13
200 CALL COLOR(I,1,1)
210 NEXT I
220 CALL COLOR(14,16,16)
230 CALL CHAR(97,"00446C5454444444")
240 CALL CHAR(98,"0038101010101038")
250 CALL CHAR(99,"0038444038044438")
260 C(1)=5
270 C(2)=7
280 C(3)=11
290 C(4)=3
300 T(1)=262
310 T(2)=294
320 T(3)=330
330 T(4)=349
340 PRINT "PRESS ANY KEY TO START.";
350 CALL KEY(0,K,ST)
360 IF ST<1 THEN 350
370 CALL CLEAR
380 CALL SCREEN(8)
390 FOR I=1 TO 4
400 CALL HCHAR(10,7*I-3,136,5)
410 CALL VCHAR(11,7*I-3,136,3)
420 CALL VCHAR(11,7*I+1,136,3)
430 CALL HCHAR(14,7*I-3,136,5)
440 CALL HCHAR(11,7*I-2,96+8*I,3)
450 CALL HCHAR(12,7*I-2,96+8*I,3)
460 CALL HCHAR(13,7*I-2,96+8*I,3)
470 CALL HCHAR(8,7*I-1,48+I)
480 NEXT I
490 CALL HCHAR(5,15,97)
500 CALL HCHAR(5,16,98)
510 CALL HCHAR(5,17,99,2)
520 RANDOMIZE
530 FOR I=1 TO 4
540 LC(I)=C(R4)
550 FOR J=1 TO I-1
560 IF LC(I)=LC(J)THEN 540
570 NEXT J
580 NEXT I
590 LF=0
600 M=0
610 CALL HCHAR(17,15,48,3)
```

```
620 LF=LF+1
630 IF LF=100 THEN 990
640 L(LF)=R4
650 FOR LL=1 TO LF
660 S=L(LL)
670 CALL SOUND(300,T(S),2)
680 CALL COLOR(S+9,LC(S),LC(S))
690 CALL SOUND(300,9999,30)
700 CALL COLOR(S+9,1,1)
710 NEXT LL
720 FOR LG=1 TO LF
730 CALL KEY(0,K,ST)
740 IF (K>48)+(K<53)<>-2 THEN 730
750 S=K-48
760 CALL SOUND(300,T(S),2)
770 CALL COLOR(S+9,LC(S),LC(S))
780 CALL SOUND(1,9999,30)
790 CALL COLOR(S+9,1,1)
800 IF S<>L(LG)THEN 900
810 NEXT LG
820 LF$="00"&STR$(LF)
830 SC$=SEG$(LF$,LEN(LF$)-2,3)
840 FOR I=1 TO LEN(SC$)
850 CALL HCHAR(17,14+I,ASC(SEG$(SC$,I,1)))
860 NEXT I
870 CALL SOUND(500,9999,30)
880 CALL SOUND(1,9999,30)
890 GOTO 620
900 CALL SOUND(500,-8,4)
910 FOR I=1 TO 20
920 CALL COLOR(9,7,1)
930 CALL COLOR(9,12,1)
940 CALL COLOR(9,2,1)
950 NEXT I
960 CALL COLOR(9,1,1)
970 M=M+1
980 IF M<3 THEN 650
990 PRINT TAB(7);"** GAME OVER **"
1000 PRINT :"PRESS 1 TO PLAY AGAIN"
1010 PRINT TAB(7);"2 TO END"
1020 CALL KEY(0,K,ST)
1030 IF K=49 THEN 370
1040 IF K<>50 THEN 1020
1050 CALL CLEAR
1060 END
```

5

Grid

C. Regena

Can you find the lost treasure? Play "Grid" and find out where in the city the treasure is. This game will also help you in creating your own guessing games.

The treasure is lost in a city where the streets are long and numerous. As with many old cities, the streets run north-south and east-west, forming a grid. Your goal is to find the lost treasure. Each time you move, the computer gives you a clue where to walk.

When the game starts, the computer prints the city streets (a grid) on the screen. The computer has hidden the treasure somewhere in the city. By using the arrow keys to position yourself (represented by the little yellow marker), move around the city until you believe you are closer to the treasure, then press ENTER. The computer gives you a clue: an arrow is printed to show you in which direction to walk. Try to find the treasure in as few guesses as possible.

Program Explanation

Use the basic idea of the game to create your own guessing game. You might choose a space hunt, a monster hidden in caves, or a bomb in a multilevel hotel. Instead of using printed arrows for clues, you might use a certain number of points for a close guess and another score for a direct hit; or you could choose different noises to indicate how close you are to the hidden object.

Lines 230-470 define graphics and colors for the game. Lines 530-630 print the labeled grid. Lines 650-670 randomly choose the point to be found, with coordinates AX and AY. Your yellow marker is BX, BY. Lines 720-1070 move the marker around until you press ENTER. Lines 1090-1190 test the point and print the appropriate clues. The subroutine in lines 1340-1390 prints the score.

Grid

```
100 REM  GRID
110 REM
```

```
120 CALL CLEAR
130 PRINT TAB(8);"***********"
140 PRINT TAB(8);"*";TAB(18);"*"
150 PRINT TAB(8);"* G R I D *"
160 PRINT TAB(8);"*";TAB(18);"*"
170 PRINT TAB(8);"***********"
180 PRINT ::::"FIND THE HIDDEN POINT"
190 PRINT "ON THE GRID."
200 PRINT :"USE THE ARROW KEYS TO MOVE"
210 PRINT "YOUR POINT TO GUESS,"
220 PRINT "THEN PRESS <ENTER>."
230 CALL CHAR(96,"0000001F1010101")
240 CALL CHAR(97,"000000FF")
250 CALL CHAR(98,"000000FF1010101")
260 CALL CHAR(99,"000000F01010101")
270 CALL CHAR(100,"101010101010101")
280 CALL CHAR(101,"1010101F1010101")
290 CALL CHAR(102,"101010FF1010101")
300 CALL CHAR(103,"101010F01010101")
310 CALL CHAR(104,"1010101F")
320 CALL CHAR(105,"101010FF")
330 CALL CHAR(106,"101010F")
340 CALL COLOR(9,5,1)
350 CALL COLOR(10,5,1)
360 A$="`abababababababac"
370 B$="d d d d d d d d d d"
380 C$="eafafafafafafafag"
390 D$="haiaiaiaiaiaiaiaj"
400 CALL CHAR(128,"3C7EFFFFFFFF7E3C")
410 CALL COLOR(13,7,1)
420 CALL CHAR(113,"0810207F201008")
430 CALL CHAR(114,"103854921010101")
440 CALL CHAR(115,"080402FF020408")
450 CALL CHAR(116,"101010109254381")
460 CALL CHAR(120,"3C7EFFFFFFFF7E3C")
470 CALL COLOR(12,12,1)
480 PRINT ::"PRESS ANY KEY TO START.";
490 CALL KEY(0,K,S)
500 IF S<1 THEN 490
510 CALL CLEAR
520 CALL SCREEN(8)
530 PRINT TAB(6);A$
540 FOR I=1 TO 7
550 PRINT TAB(6);B$:TAB(6);C$
560 NEXT I
570 PRINT TAB(6);B$;TAB(6);D$
580 C=48
590 PRINT TAB(6);"0 1 2 3 4 5 6 7 8 9"
600 FOR I=22 TO 6 STEP -2
```

```
610 CALL HCHAR(I,7,C)
620 C=C+1
630 NEXT I
640 PRINT ::TAB(10);"SCORE:"
650 RANDOMIZE
660 AX=INT(RND*10)
670 AY=INT(RND*9)
680 SC=0
690 G=104
700 BX=0
710 BY=0
720 CALL GCHAR(19,8,G)
730 I=19-BY*2
740 J=8+BX*2
750 CALL HCHAR(I,J,120)
760 CALL KEY(0,K,S)
770 IF K=13 THEN 1080
780 IF K<>69 THEN 840
790 DI=-1
800 DJ=0
810 IF BY+1<9 THEN 1010
820 DI=0
830 GOTO 1010
840 IF K<>68 THEN 900
850 DI=0
860 DJ=1
870 IF BX+1<10 THEN 1010
880 DJ=0
890 GOTO 1010
900 IF K<>83 THEN 960
910 DI=0
920 DJ=-1
930 IF BX-1>-1 THEN 1010
940 DJ=0
950 GOTO 1010
960 IF K<>88 THEN 760
970 DI=1
980 DJ=0
990 IF BY-1>-1 THEN 1010
1000 DI=0
1010 CALL HCHAR(I,J,G)
1020 I=I+2*DI
1030 J=J+2*DJ
1040 CALL GCHAR(I,J,G)
1050 BX=BX+DJ
1060 BY=BY-DI
1070 GOTO 750
1080 CALL SOUND(100,-1,2)
1090 IF (AX=BX)+(AY=BY)=-2 THEN 1230
```

```
1100 IF AX<=BX THEN 1130
1110 CALL HCHAR(19-BY*2,8+BX*2,115)
1120 GOTO 1200
1130 IF AX>=BX THEN 1160
1140 CALL HCHAR(19-BY*2,8+BX*2,113)
1150 GOTO 1200
1160 IF AY>=BY THEN 1190
1170 CALL HCHAR(19-BY*2,8+BX*2,116)
1180 GOTO 1200
1190 CALL HCHAR(19-BY*2,8+BX*2,114)
1200 GOSUB 1340
1210 CALL SOUND(100,1497,2)
1220 GOTO 700
1230 CALL HCHAR(19-AY*2,8+AX*2,128)
1240 GOSUB 1340
1250 FOR I=1 TO 15
1260 CALL SOUND(-100,INT(200*RND+200),2)
1270 CALL COLOR(13,12,1)
1280 CALL COLOR(13,7,1)
1290 NEXT I
1300 PRINT :"TRY AGAIN? (Y/N)";
1310 CALL KEY(0,K,S)
1320 IF K=89 THEN 510
1330 IF K=78 THEN 1400 ELSE 1310
1340 SC=SC+1
1350 S$=STR$(SC)
1360 FOR I=1 TO LEN(S$)
1370 CALL HCHAR(23,18+I,ASC(SEG$(S$,I,1)))
1380 NEXT I
1390 RETURN
1400 CALL CLEAR
1410 END
```

5

Wordsearch

C. Regena

Since you enter your own words, this game is suitable for both beginning readers and those who have enjoyed puzzles for years.

This program creates a "Wordsearch" puzzle like those found in magazines and books. You may enter up to 20 words. The words are placed in the puzzle horizontally, vertically, or diagonally. Random letters fill the remaining spaces, then the puzzle is printed. Your object is to find the hidden words.

Program Explanation

This program yields a puzzle 20 letters by 20 letters. Initially, all spots are filled with dots. As a word is input, a random direction and starting position are chosen, then the letters of the word are put into the M$(20,20) array. After the words are entered, lines 1170-1220 fill the puzzle with random letters wherever dots were. Lines 1240-1290 print the puzzle.

If a word cannot be filled in, you have your choice of trying another word, starting the puzzle all over, or stopping to print the puzzle as it is without adding more words.

Variations

You may want to let the user input values for the number of rows and number of columns. If so, change all values of 20 to the variable names in lines 670-780.

If you have a printer and would like the puzzle printed, add the lines below (using your own printer configuration in line 1300). Following these lines is a sample puzzle printed by the program. Try to find the names of ten computers in the puzzle.

```
1300 OPEN #1:"RS232.BA=600"
1310 FOR A=1 TO 20
1320 FOR B=1 TO 20
1330 PRINT #1:M$(A,B);" ";
1340 NEXT B
1350 PRINT #1
```

```
1360 NEXT A
1370 CLOSE #1
1380 END
```

Sample Puzzle Generated by "Wordsearch"

```
O H A V U Q G F I S E J S U G X R Y K W
O G T C Z I G B S R B A U U I O K Y X T
Y Y G F C N Z Z H M Z A Z R I S H F N E
Z F U C G A D X K R L G A P F V A H L X
G E W B J Z A V A A A T V I C Q Y C C A
Y K L H B G J I V H A G Z Y I L X O J S
Q Y H C D R N Q Z G S O N Y E O C M I I
S K A P C B D T C A Z M X N A L M M I N
H K Q K O P Q C Y I G Z R J A Q S O D S
R R B W I T G X N F F O Q C T E K D E T
F E Z J L B Q I V S B G Z G J C V O O R
L N S V A S L Z V S L R O G A O A R O U
O F F Y G K K C O E I L J H X B E E U M
Z H V M N T L E L A H Z S C L P Q T C E
H B J A V A K P L K B O W N H R A N Z N
V W R U L W P C V U I X V J B G M F J T
J F X H B A N O J D S J M L M H R N D S
C V W T F I R E A Z H W N J Q Q S R S C
U G N Z S W G R U I N O X A Z W U V G G
L R J B V M P X S T T X I S X H O Y X X
```

Wordsearch

```
100 REM  WORDSEARCH
110 DIM L$(26),N$(20),M$(20,20)
120 CALL CLEAR
130 PRINT TAB(9);"WORDSEARCH"::::
140 PRINT "YOU MAY ENTER UP TO TWENTY"
150 PRINT :"WORDS.  THEY WILL BE"
160 PRINT :"PLACED HORIZONTALLY,"
170 PRINT :"VERTICALLY, OR DIAGONALLY"
180 PRINT :"IN A WORDSEARCH PUZZLE.":::
190 FOR C=1 TO 26
200 L$(C)=CHR$(64+C)
210 NEXT C
220 FOR A=1 TO 20
230 FOR C=1 TO 20
240 M$(A,C)="."
```

```
250 NEXT C
260 NEXT A
270 INPUT "HOW MANY WORDS? ":W
280 IF W<21 THEN 310
290 PRINT :"SORRY 20 OR LESS.":
300 GOTO 270
310 FOR C=1 TO W
320 INPUT "WORD: ":N$(C)
330 A=0
340 L=LEN(N$(C))
350 IF L<20 THEN 380
360 PRINT :"SORRY, MUST BE SHORTER.":
370 GOTO 320
380 A=A+1
390 IF A<100 THEN 490
400 PRINT :"COULDN'T FIT WORD IN."
410 PRINT :"CHOOSE:  1 START ALL OVER"
420 PRINT TAB(10);"2 TRY ANOTHER WORD"
430 PRINT TAB(10);"3 STOP NOW"
440 CALL SOUND(150,1497,2)
450 CALL KEY(0,K,S)
460 IF (K<49)+(K>51)THEN 450
470 CALL CLEAR
480 ON K-48 GOTO 220,320,1160
490 D=1
500 E=1
510 F=-1
520 RANDOMIZE
530 IF RND<=.5 THEN 550
540 F=1
550 G=-1
560 IF RND<=.5 THEN 580
570 G=1
580 H=2
590 IF F<>1 THEN 610
600 D=0
610 IF G<>1 THEN 630
620 E=0
630 IF RND<.75 THEN 650
640 H=1
650 IF RND>.25 THEN 670
660 H=0
670 COL=20
680 ROW=20
690 IF H<>1 THEN 710
700 ROW=20-L
710 IF H<>0 THEN 730
720 COL=20-L
730 IF H<=1 THEN 760
```

```
740 ROW=20-L
750 COL=20-L
760 IF COL<>20 THEN 780
770 E=0
780 IF ROW<>20 THEN 800
790 D=0
800 II=INT(ROW*RND+1)
810 IJ=INT(COL*RND+1)
820 I=II+D*L
830 J=IJ+E*L
840 IF H=1 THEN 1060
850 IF H=0 THEN 960
860 FOR P=1 TO L
870 IF M$(J+(P-1)*G,I+(P-1)*F)="." THEN 890
880 IF M$(J+(P-1)*G,I+(P-1)*F)<>SEG$(N$(C),P
    ,1)THEN 380
890 NEXT P
900 A=0
910 FOR P=1 TO L-1
920 M$(J+P*G,I+P*F)=SEG$(N$(C),P+1,1)
930 NEXT P
940 M$(J,I)=SEG$(N$(C),1,1)
950 GOTO 1150
960 FOR P=1 TO L
970 IF M$(J+(P-1)*G,I)="." THEN 990
980 IF M$(J+(P-1)*G,I)<>SEG$(N$(C),P,1)THEN
    380
990 NEXT P
1000 A=0
1010 FOR P=1 TO L-1
1020 M$(J+P*G,I)=SEG$(N$(C),P+1,1)
1030 NEXT P
1040 M$(J,I)=SEG$(N$(C),1,1)
1050 GOTO 1150
1060 FOR P=1 TO L
1070 IF M$(J,I+(P-1)*F)="." THEN 1090
1080 IF M$(J,I+(P-1)*F)<>SEG$(N$(C),P,1)THEN
      380
1090 NEXT P
1100 A=0
1110 FOR P=1 TO L-1
1120 M$(J,I+P*F)=SEG$(N$(C),P+1,1)
1130 NEXT P
1140 M$(J,I)=SEG$(N$(C),1,1)
1150 NEXT C
1160 PRINT :"ONE MOMENT PLEASE."
1170 FOR A=1 TO 20
1180 FOR B=1 TO 20
1190 IF M$(A,B)<>"." THEN 1210
```

```
1200 M$(A,B)=L$(INT(26*RND+1))
1210 NEXT B
1220 NEXT A
1230 CALL CLEAR
1240 FOR A=1 TO 20
1250 FOR B=1 TO 20
1260 PRINT M$(A,B);
1270 NEXT B
1280 PRINT
1290 NEXT A
1300 END
```

5

MathMan

Andy Hayes
Translated for the TI by C. Regena

Here's proof that computer-aided math practice need not be boring.

"MathMan" is a multiplication drill program for children of elementary school age. Type in the answer to the random multiplication problem shown. If the answer is correct, the friends gathered will cheer. If the answer is incorrect, the correct answer is shown, but one of the friends will leave. When all six friends have gone, the game is over.

You may advance to the next level by answering ten problems correctly. The problems get successively more difficult, and the score for each problem also increases with each level.

Program Explanation

Lines 250-510 define graphics characters and colors. Lines 550-570 initialize variables for each new game. The variable A is the number of friends, LV is the level number, and SC is the score. Lines 600-680 print the platform and an invisible ray (the lowercase p). If an answer is incorrect, the "p" ray lights up. Lines 690-750 draw the correct number of friends for each level. Lines 1310-1330 erase one of the friends.

The problems chosen and the scoring depend on the level number. Lines 800-810 choose the problem. The top number may be a number from one to two times the level number. The second number may be a number from one to nine. The answer is BC. The score for each problem is five times the level number—for Level 1 each problem is worth 5 points; for Level 2 each problem is worth 10 points, etc.

MathMan

```
100 REM  MATH MAN
110 CALL CLEAR
120 PRINT TAB(7);"************"
130 PRINT TAB(7);"* MATH MAN *"
140 PRINT TAB(7);"************"
150 PRINT :::"TYPE IN THE CORRECT ANSWER."
```

```
160 PRINT :"IF YOU ARE CORRECT, YOUR"
170 PRINT "FRIENDS GATHERED BELOW"
180 PRINT "WILL CHEER."
190 PRINT :"ANSWER 10 CORRECTLY TO"
200 PRINT "GO TO THE NEXT LEVEL."
210 PRINT :"IF YOU MISS, ONE FRIEND"
220 PRINT "WILL LEAVE."
230 PRINT :"GAME ENDS WHEN ALL SIX"
240 PRINT "ARE GONE.":::
250 CALL CHAR(95,"000000FF")
260 CALL CHAR(104,"387CFEFEFE7C38FE")
270 CALL CHAR(105,"0F0F")
280 CALL CHAR(106,"FFFF7C7C7C7CFEFE")
290 CALL CHAR(107,"E0E")
300 CALL CHAR(108,"FEEEEEEEEEEEEEE")
310 CALL COLOR(10,5,1)
320 CALL CHAR(120,"0038444040404438")
330 CALL CHAR(121,"007C44444444447C")
340 CALL CHAR(122,"0078444478504844")
350 CALL CHAR(123,"007C40407840407C")
360 CALL CHAR(124,"007C101010101010")
370 CALL COLOR(12,1,1)
380 CALL COLOR(9,12,12)
390 CALL CHAR(112,"0102040810204080")
400 CALL COLOR(11,1,1)
410 FOR T=128 TO 138
420 READ C$
430 CALL CHAR(T,C$)
440 NEXT T
450 DATA 0038444444444438,0010301010101038
460 DATA 003844040810207C,0038440418044438
470 DATA 00081828487C0808,007C407804044438
480 DATA 0018204078444438,007C040810202020
490 DATA 0038444438444438,003844443C040830,0
500 CALL COLOR(14,2,11)
510 CALL COLOR(13,2,11)
520 PRINT "PRESS ANY KEY TO START."
530 CALL KEY(0,K,ST)
540 IF ST<1 THEN 530
550 A=6
560 LV=1
570 SC=0
580 CALL CLEAR
590 PRINT "LEVEL ";LV::::::::::
600 FOR T=13 TO 9 STEP -1
610 PRINT TAB(T);"p"
620 NEXT T
630 PRINT "``{5 SPACES}p{7 SPACES}xyzz{x|"
640 PRINT "```{3 SPACES}p"
```

```
650 PRINT "`````p"
660 PRINT "``````"
670 PRINT "```````"
680 PRINT "````````"
690 FOR I=1 TO A
700 CALL HCHAR(21,30-I*3,104)
710 CALL HCHAR(22,29-I*3,105)
720 CALL HCHAR(22,30-I*3,106)
730 CALL HCHAR(22,31-I*3,107)
740 CALL HCHAR(23,30-I*3,108)
750 NEXT I
760 PRINT "SCORE=";SC::
770 S=LV*2
780 O=0
790 RANDOMIZE
800 B=INT(RND*S)+1
810 C=INT(RND*9)+1
820 B$=STR$(B)
830 FOR I=1 TO LEN(B$)
840 CALL HCHAR(6,16-LEN(B$)+I,ASC(SEG$(B$,I,1)))
850 NEXT I
860 CALL HCHAR(8,14,88)
870 CALL HCHAR(8,16,C+48)
880 CALL HCHAR(9,14,95,4)
890 BC=B*C
900 BC$=STR$(BC)
910 L=LEN(BC$)
920 AS$=""
930 FOR I=1 TO L
940 CALL KEY(0,K,ST)
950 CALL HCHAR(10,16-L+I,63)
960 CALL HCHAR(10,16-L+I,32)
970 IF (K<48)+(K>57)THEN 940
980 CALL HCHAR(10,16-L+I,K)
990 AS$=AS$&STR$(K-48)
1000 NEXT I
1010 AS=VAL(AS$)
1020 IF AS<>BC THEN 1160
1030 CALL COLOR(12,7,16)
1040 CALL SOUND(100,262,2)
1050 CALL SOUND(100,330,2)
1060 CALL SOUND(100,392,2)
1070 CALL SOUND(200,523,2)
1080 SC=SC+5*LV
1090 SC$=STR$(SC)
1100 FOR J=1 TO LEN(SC$)
1110 CALL HCHAR(22,9+J,ASC(SEG$(SC$,J,1)))
1120 NEXT J
1130 CALL COLOR(12,1,1)
```

```
1140 O=O+1
1150 GOTO 1380
1160 CALL SOUND(500,-8,2)
1170 FOR D=1 TO 200
1180 NEXT D
1190 CALL COLOR(11,7,1)
1200 CALL COLOR(11,1,1)
1210 CALL HCHAR(10,15,138,3)
1220 CALL VCHAR(9,16,138,3)
1230 CALL HCHAR(9,16,95)
1240 CALL HCHAR(10,15,32,3)
1250 CALL HCHAR(11,16,32)
1260 CALL COLOR(11,7,1)
1270 FOR J=1 TO L
1280 CALL HCHAR(10,16-L+J,ASC(SEG$(BC$,J,1))
     +80)
1290 NEXT J
1300 CALL COLOR(11,1,1)
1310 FOR J=29-A*3 TO 29-A*3+2
1320 CALL VCHAR(19,J,32,3)
1330 NEXT J
1340 FOR D=1 TO 500
1350 NEXT D
1360 A=A-1
1370 IF A=0 THEN 1540
1380 CALL HCHAR(6,3,32,15)
1390 CALL HCHAR(8,3,32,15)
1400 CALL HCHAR(10,3,32,15)
1410 IF O<10 THEN 790
1420 FOR D=1 TO 500
1430 NEXT D
1440 CALL CLEAR
1450 PRINT "YOU MADE IT THROUGH"
1460 PRINT :"LEVEL";LV
1470 LV=LV+1
1480 IF LV=11 THEN 1570
1490 PRINT ::"YOU NOW ADVANCE TO"
1500 PRINT :"LEVEL";LV
1510 PRINT :::"PRESS ANY KEY TO START."
1520 CALL KEY(0,K,ST)
1530 IF ST=1 THEN 580 ELSE 1520
1540 PRINT "SORRY BUT ALL YOUR"
1550 PRINT "FRIENDS HAVE LEFT"
1560 PRINT :"LEVEL ";LV
1570 PRINT :::::"TRY AGAIN? (Y/N)"
1580 CALL KEY(0,K,ST)
1590 IF K=89 THEN 550
1600 IF K<>78 THEN 1580
1610 CALL CLEAR
1620 END
```

5

Memory Trainer

Harvey B. Herman
Translated for the TI by Patrick Parrish

This program might help you improve your memory skills. Some people, training in a similar fashion, have been able to quickly memorize random 80-digit numbers.

A provocative article, entitled "Exceptional Memory", appeared recently in *American Scientist* (vol. 70, no. 6, p. 607, 1982). The authors described experiments in which a person with a normal memory was trained to recall a sequence of over 80 random digits. How?

When most people read a random sequence once, they can remember only five to nine digits, the apparent limit of short-term memory (STM).

One might call this prodigious feat of memory (recalling 80 digits) *exceptional*, but the authors said that this skill may not be uncommon. Diligent practice, in one case 230 hours over 20 months, resulted in improvement in the ability to rapidly transfer information into long-term memory (LTM). A "normal" memory could thereby be transformed into an "exceptional" one.

After reading the article, I realized how easy it would be to computerize the memory-training task. Before long, I had written "Memory Trainer."

Random digits are flashed on the screen at a specified rate, rather than being read to the subject. If the sequence is repeated correctly, the next sequence of digits is increased by one. When an error is made, the length of the sequence decreases by one. The subject can stop the experiment at any point, whereupon the maximum number length achieved is displayed.

Memory Trainer

```
100 RANDOMIZE
110 REM  MAX 90 DIGITS
120 DIM N(90)
130 MA=0
140 CALL CLEAR
150 CALL SCREEN(12)
```

```
160 FOR I=5 TO 8
170 CALL COLOR(I,14,16)
180 NEXT I
190 PRINT "{6 SPACES}MEMORY TRAINER"
200 FOR I=1 TO 3
210 PRINT
220 NEXT I
230 INPUT "DIGIT RATE (1-10) ? ":DR
240 IF (DR<1)+(DR>10)THEN 140
250 PRINT
260 INPUT "INITIAL SEQUENCE LENGTH ? ":SL
270 IF SL<2 THEN 280 ELSE 290
280 SL=2
290 IF SL>90 THEN 300 ELSE 310
300 SL=90
310 PRINT
320 PRINT
330 PRINT
340 PRINT "CURRENT DIGIT SPAN "&STR$(SL)
350 PRINT
360 PRINT
370 IF H$<>"Y" THEN 410
380 FOR I=1 TO 6
390 PRINT
400 NEXT I
410 PRINT "get set"
420 PRINT "*"
430 FOR I=9 TO 11
440 CALL COLOR(I,10,7)
450 NEXT I
460 FOR I=1 TO 200
470 NEXT I
480 FOR I=9 TO 11
490 CALL COLOR(I,2,1)
500 NEXT I
510 CALL SOUND(150,300,10)
520 FOR I=1 TO 200
530 NEXT I
540 FOR I=1 TO SL
550 N(I)=INT(RND*10)
560 CALL HCHAR(23,3,N(I)+48)
570 FOR J=1 TO DR*20
580 NEXT J
590 CALL HCHAR(23,3,32)
600 FOR K=1 TO 10
610 NEXT K
620 NEXT I
630 CALL HCHAR(23,3,32)
640 FL=0
```

```
650 PRINT
660 PRINT
670 IF H$<>"Y" THEN 710
680 FOR I=1 TO 4
690 PRINT
700 NEXT I
710 PRINT "INPUT DIGITS"
720 INPUT "*":A$
730 PRINT
740 PRINT
750 FOR I=1 TO 200
760 NEXT I
770 IF LEN(A$)=SL THEN 800
780 FL=1
790 GOTO 870
800 FOR I=1 TO SL
810 IF VAL(SEG$(A$,I,1))=N(I)THEN 840
820 I=SL
830 FL=1
840 NEXT I
850 REM  FL=0 -CORRECT- INCREASE SEQ LEN BY
    ONE
860 REM  FL=1 -INCORRECT-
870 CALL CLEAR
880 IF FL=0 THEN 1000
890 PRINT "INCORRECT-TRY A SHORTER SPAN"
900 PRINT
910 SL=SL-1
920 PRINT "YOUR RESPONSE=";A$
930 H$=""
940 FOR I=1 TO SL+1
950 H$=H$&STR$(N(I))
960 NEXT I
970 PRINT
980 PRINT "ACTUAL SEQUENCE=";H$
990 GOTO 1040
1000 PRINT " CORRECT-TRY A LONGER SPAN"
1010 SL=SL+1
1020 IF MA>=SL-1 THEN 1040
1030 MA=SL-1
1040 FOR I=1 TO 10
1050 PRINT
1060 NEXT I
1070 PRINT "{6 SPACES}AGAIN (Y OR N) ? "
1080 CALL KEY(0,F,STATUS)
1090 IF STATUS=0 THEN 1080
1100 H$=CHR$(F)
1110 CALL CLEAR
1120 IF H$="N" THEN 1140
```

```
1130 GOTO 340
1140 PRINT "HOPE YOU IMPROVED YOUR SPAN!"
1150 FOR I=1 TO 10
1160 PRINT
1170 NEXT I
1180 FOR I=3 TO 4
1190 CALL COLOR(I,13,12)
1200 NEXT I
1210 PRINT "-HIGHEST CORRECT DIGIT SPAN-"
1220 PRINT
1230 PRINT
1240 PRINT "{11 SPACES}(";MA;")"
1250 FOR I=1 TO 500
1260 NEXT I
1270 END
```

6

Creative Games

6

Make a Face

C. Regena

This simple program helps teach young children how to make menu selections while allowing them to be creative.

Very few computer programs allow young children to be creative. Most programs tell children what to do. Sure, they are given the options usually present in computer games and programs, but rarely are they allowed to be truly creative.

Enter "Make a Face." The TI will act as a face-maker. A child may choose from several options by pressing a number. What kind of choices? A nose, a mouth, eyes, and eyebrows.

At the end, the child has a face he or she has created. If the child wishes to create another face, he or she just presses Y when asked.

Program Explanation

Lines 170-280 define the graphics characters for the facial features. Care was taken to define the graphics in a certain order so later graphics characters can be defined by a variable in terms of the number chosen (lines 460, 660, 780, 960). Using the variable G, the choices are drawn first. After the child chooses a number, the appropriate feature is drawn on the face.

Make a Face

```
120 CALL CLEAR
130 PRINT TAB(8);"MAKE A FACE"
140 PRINT :::"CHOOSE EYES, EYEBROWS,"
150 PRINT :"NOSE, AND MOUTH BY PRESSING"
160 PRINT :"THE CORRESPONDING NUMBER.":::::
170 FOR C=96 TO 147
180 READ C$
190 CALL CHAR(C,C$)
200 NEXT C
210 DATA 2424242424424242,424242428181817E,0
    20404080810102,20404080808C936,,18244281
    81422481,0
220 DATA 18242442428181FF,000000073B274F8F,0
    00000E0D8E4F2F1,4F2F170F03,F2F4E8F0C,0F1
    0204080838078F
```

6

```
230 DATA F008040201C1E1F1,4F4F4F2F2F2F708,F2
    F2F2F4F4F40E01,00030C1020214347,00C03008
    0484C2E2
240 DATA 47432120100C03,E2C284040830C,000000
    00000F7,0000031CE,0000C03807,0000000000F
    00E,00000000030C30C
250 DATA 00000000C02418,00000000032418,00000
    000C0300C03,000071860808,0000E018060101
    ,000071860808
260 DATA 0000E018060101,0102040808040201,FF0
    00000000000FF,804020101040208,030F3FFF7F
    1F07,C7EFFFFFFFFFFFE
270 DATA 80E0F8FEFCF0C,2040A010080601,000000
    000000817E,0402050810608,8060783F1F0F03,
    00000000FFFFFFFF
280 DATA 01061EFCF8F0C,,,,,FF,01020408102040
    8,0101010101010101,8040201008040201
290 CALL COLOR(13,7,1)
300 CALL COLOR(14,7,1)
310 CALL COLOR(15,9,1)
320 CALL CLEAR
330 CALL HCHAR(4,20,144,5)
340 CALL VCHAR(7,16,146,6)
350 CALL VCHAR(7,27,146,6)
360 CALL HCHAR(17,21,144,3)
370 RESTORE 380
380 DATA 4,19,145,4,25,147,5,18,145,5,26,147
    ,6,17,145,6,27,147,13,17,147,13,27,145
390 DATA 14,18,147,14,26,145,15,19,147,15,25
    ,145,16,20,147,16,24,145
400 DATA 5,3,67,5,4,72,5,5,79,5,6,79,5,7,83,
    5,8,69,5,9,58,8,3,49,12,3,50,16,3,51
410 FOR C=1 TO 24
420 READ A,B,G
430 CALL HCHAR(A,B,G)
440 NEXT C
450 FOR C=1 TO 3
460 G=100+C*4
470 CALL HCHAR(4+C*4,5,G)
480 CALL HCHAR(4+C*4,6,G+1)
490 CALL HCHAR(5+C*4,5,G+2)
500 CALL HCHAR(5+C*4,6,G+3)
510 NEXT C
520 CALL SOUND(150,1497,2)
530 CALL KEY(0,K,S)
540 IF (K<49)+(K>51)THEN 530
550 G=100+(K-48)*4
560 FOR C=20 TO 23 STEP 3
570 CALL HCHAR(8,C,G)
```

```
580 CALL HCHAR(8,C+1,G+1)
590 CALL HCHAR(9,C,G+2)
600 CALL HCHAR(9,C+1,G+3)
610 NEXT C
620 CALL VCHAR(8,5,32,10)
630 CALL VCHAR(8,6,32,10)
640 CALL HCHAR(20,3,52)
650 FOR C=1 TO 4
660 G=94+2*C
670 CALL HCHAR(4+C*4,5,G)
680 CALL HCHAR(5+C*4,5,G+1)
690 NEXT C
700 CALL SOUND(150,1497,2)
710 CALL KEY(0,K,S)
720 IF (K<49)+(K>52)THEN 710
730 C=K-48
740 CALL HCHAR(10,22,94+2*C)
750 CALL HCHAR(11,22,95+2*C)
760 CALL VCHAR(8,5,32,14)
770 FOR C=1 TO 4
780 G=C*3+125
790 CALL HCHAR(4+C*4,5,G)
800 CALL HCHAR(4+C*4,6,G+1)
810 CALL HCHAR(4+C*4,7,G+2)
820 NEXT C
830 CALL SOUND(150,1497,2)
840 CALL KEY(0,K,S)
850 IF (K<49)+(K>52)THEN 840
860 C=K-48
870 G=125+C*3
880 CALL HCHAR(13,21,G)
890 CALL HCHAR(13,22,G+1)
900 CALL HCHAR(13,23,G+2)
910 CALL VCHAR(8,5,32,13)
920 CALL VCHAR(8,6,32,13)
930 CALL VCHAR(8,7,32,13)
940 CALL HCHAR(20,3,32)
950 FOR C=1 TO 3
960 G=112+C*4
970 CALL HCHAR(4+C*4,5,G)
980 CALL HCHAR(4+C*4,6,G+1)
990 NEXT C
1000 CALL SOUND(150,1497,2)
1010 CALL KEY(0,K,S)
1020 IF (K<49)+(K>51)THEN 1010
1030 C=K-48
1040 G=112+C*4
1050 CALL HCHAR(6,20,G)
1060 CALL HCHAR(6,21,G+1)
```

```
1070 CALL HCHAR(6,23,G+2)
1080 CALL HCHAR(6,24,G+3)
1090 CALL VCHAR(8,5,32,9)
1100 CALL VCHAR(8,6,32,9)
1110 CALL VCHAR(5,3,32,12)
1120 CALL HCHAR(5,4,32,6)
1130 PRINT "TRY AGAIN? (Y/N)";
1140 CALL KEY(0,K,S)
1150 IF K=89 THEN 320
1160 IF K<>78 THEN 1140
1170 CALL CLEAR
1180 END
```

Joystick Drawing

C. Regena

This simple drawing game shows how to program for both joystick and keyboard input.

You may use either the standard arrow keys or joystick number 1 to make a design. Move the joystick or press an arrow key to draw in the desired direction. To change colors, press the fire button or the Q key. One of the colors is the background color or a blank. If you get lost, simply press the fire button or Q key to see where you are.

Program Explanation

Setting the foreground and background colors the same in each of seven color sets also defines the colored squares. G is the character number of the symbol being drawn—the first character number in each color set. Lines 340-550 check the keyboard and the joystick alternately. If the Q key or the fire button has been pressed, the procedure in lines 570-600 sounds a short beep and changes the color by determining a different graphic character number.

Joystick Drawing

```
100 REM  JOYSTICK DRAWING
110 REM
120 CALL CLEAR
130 PRINT TAB(6);"JOYSTICK DRAWING"
140 PRINT :::"{3 SPACES}PRESS ARROW KEYS OR
    "
150 PRINT :"{3 SPACES}USE JOYSTICKS TO DRAW
    ."
160 PRINT ::"{3 SPACES}PRESS 'Q' OR THE FIR
    E"
170 PRINT :"{3 SPACES}BUTTON TO CHANGE COLO
    RS."
180 CALL COLOR(3,3,3)
190 CALL COLOR(4,16,16)
200 I=15
210 J=12
```

```
220 RANDOMIZE
230 G=INT(RND*7+1)*8+32
240 PRINT ::::"{3 SPACES}PRESS ANY KEY TO S
    TART."
250 CALL KEY(0,K,S)
260 IF S<1 THEN 250
270 CALL CLEAR
280 CALL COLOR(2,2,2)
290 CALL COLOR(5,5,5)
300 CALL COLOR(6,14,14)
310 CALL COLOR(7,7,7)
320 CALL COLOR(8,12,12)
330 CALL SCREEN(8)
340 CALL HCHAR(J,I,G)
350 CALL KEY(1,K,S)
360 IF S=0 THEN 500
370 IF K=18 THEN 570
380 IF K>5 THEN 350
390 II=0
400 JJ=0
410 ON K+1 GOTO 420,350,440,460,350,480
420 JJ=-4
430 GOTO 530
440 II=-4
450 GOTO 520
460 II=4
470 GOTO 520
480 JJ=4
490 GOTO 520
500 CALL JOYST(1,II,JJ)
510 IF (II=0)+(JJ=0)=-2 THEN 350
520 I=I+II/4
530 J=J-JJ/4
540 I=INT(32*((I-1)/32-INT((I-1)/32)))+1
550 J=INT(24*((J-1)/24-INT((J-1)/24)))+1
560 GOTO 340
570 CALL SOUND(100,1497,2)
580 G=G+8
590 IF G<89 THEN 340
600 G=32
610 GOTO 340
620 END
```

7

Scrolling

7

River Trip

C. Regena

This game uses the natural scrolling of the screen to create good movement.

Watch out for the rocks. Beware of fallen trees. And be sure not to run into the shore. White-water rafting trips can be fun, but you have to know what you are doing.

"River Trip" takes advantage of the natural scrolling created by constantly printing to the screen. You are in charge of the river trip. Your responsibility is to maneuver the raft safely to the end of its journey.

That yellow face is you in your raft. Use the left and right arrow keys to try to keep the raft on the river. Sound easy? The river is constantly meandering, and there are other obstacles to avoid.

Program Explanation

This program should be just the beginning. Use your own theme and design your own graphics to create your own version. You might choose a car or truck driving down the road, or you might draw a skier who must avoid trees as he slaloms down the hill. Perhaps you could design a monster chasing an innocent victim.

A$ in line 250 is the path — characters which have been redefined purple and white. Line 380 chooses J1 randomly to be 0, 1, 2, or 3 columns to the left or to the right of the previous step. Lines 390-450 make sure that you don't get a straight path at one edge and that A$ will be within the correct margins to be printed. Line 460 prints the next step of the path.

Lines 480-590 detect which arrow key was pressed, move the face two columns to the right or to the left, and erase the old position.

River Trip

```
100 REM{3 SPACES}RIVER TRIP
110 REM
120 CALL CLEAR
130 PRINT "R":"  I":TAB(5);"V":TAB(7);"E"
140 PRINT TAB(9);"R"
```

```
150 PRINT TAB(14);"T R I P"
160 PRINT :::"TRY TO STAY ON THE PATH BY"
170 PRINT :"PRESSING THE LEFT AND"
180 PRINT :"RIGHT ARROW KEYS."
190 CALL CHAR(96,"3C7EDBFFDBE77E3C")
200 CALL CHAR(97,"E7E738381C1CE7E7")
210 CALL COLOR(9,12,16)
220 CALL COLOR(10,16,14)
230 CALL CHAR(104,"FFFFFFFFFFFFFFFF")
240 CALL CHAR(105,"0")
250 A$="iihhhhhhhii"
260 PRINT ::::"PRESS ANY KEY TO START.";
270 CALL KEY(0,K,S)
280 IF S<1 THEN 270
290 CALL CLEAR
300 RANDOMIZE
310 X=19
320 G=104
330 FOR J=10 TO 15
340 PRINT TAB(J);A$
350 NEXT J
360 CALL HCHAR(22,X,96)
370 FOR I=1 TO 50
380 J1=(-1)^(INT(4*RND))*(INT(4*RND))
390 IF (J+J1=J2)+((J+J1)>=20)THEN 380
400 J=J+J1
410 J2=J
420 IF J<18 THEN 440
430 J=18
440 IF J>5 THEN 460
450 J=5
460 PRINT I;TAB(J);A$
470 CALL SOUND(-100,440,4)
480 CALL HCHAR(21,X,G)
490 CALL KEY(0,K,S)
500 IF (K<>83)+(K<>68)=-2 THEN 590
510 IF K=83 THEN 560
520 X=X+2
530 IF X<31 THEN 590
540 X=31
550 GOTO 590
560 X=X-2
570 IF X>2 THEN 590
580 X=2
590 CALL GCHAR(22,X,G)
600 IF G=104 THEN 640
610 CALL SOUND(-50,-5,0)
620 CALL HCHAR(22,X,97)
630 GOTO 690
```

```
640 CALL HCHAR(22,X,96)
650 NEXT I
660 FOR I=1 TO 20
670 CALL SOUND(100,INT(RND*500)+500,2)
680 NEXT I
690 PRINT ::"TRY AGAIN? (Y/N)"
700 CALL KEY(0,K,S)
710 IF K=89 THEN 290
720 IF K<>78 THEN 700
730 CALL CLEAR
740 END
```

7

Across

C. Regena

The challenging game "Across" is another example of how scrolling can be used in games.

The obstacles are in place. The course is ready. Are you?

Your goal in "Across" is to run successfully from one end of the obstacle course to the other. Moving from left to right across the screen, you try to avoid running into any barriers.

If you press any key, you will move to the right one spot and down one spot. You cannot back up. If you do not press a key, you will continue downward.

Program Explanation

This program uses the computer's scrolling feature to move the barriers upward. The level of difficulty may be a number from one to nine, then that variable, L, determines how many barriers are drawn on each horizontal line. Lines 410-420 determine whether to move the runner or not. If no key is pressed, the status variable is zero and the barrier pattern scrolls. If any key is pressed, the runner moves one position to the right—and the scrolling moves the runner down one. CALL GCHAR determines if there is a barrier in the runner's path.

Across

```
100 REM  ACROSS
110 REM
120 CALL CLEAR
130 PRINT TAB(7);"** ACROSS **"
140 BS=99999
150 PRINT ::::"  PRESS ANY KEY TO MOVE."
160 PRINT :"  GO TO THE RIGHT EDGE AS"
170 PRINT :"  SOON AS POSSIBLE WITHOUT"
180 PRINT :"  BUMPING ANY BARRIERS."
190 CALL CHAR(96,"3838107C10284282")
200 CALL CHAR(104,"FFFFFFFFFFFFFFFF")
210 CALL CHAR(105,"83E22618186447C1")
220 CALL COLOR(10,7,1)
230 PRINT :::"  PRESS <ENTER> TO START."
240 CALL KEY(0,K,S)
```

```
250 IF K<>13 THEN 240
260 CALL CLEAR
270 X=12
280 Y=3
290 SC=0
300 PRINT "LEVEL OF DIFFICULTY? (1-9)"
310 CALL SOUND(150,1497,2)
320 CALL KEY(0,K,S)
330 IF (K<49)+(K>57)THEN 320
340 L=K-48
350 CALL CLEAR
360 RANDOMIZE
370 FOR I=1 TO INT(RND*L+1)
380 CALL HCHAR(24,INT(RND*28+3),104)
390 NEXT I
400 SC=SC+1
410 CALL KEY(0,K,S)
420 IF S=0 THEN 480
430 IF X+1<25 THEN 450
440 X=23
450 CALL GCHAR(X+1,Y+1,G)
460 Y=Y+1
470 IF G=104 THEN 650
480 PRINT
490 IF X<25 THEN 510
500 X=24
510 CALL GCHAR(X,Y,G)
520 IF G=104 THEN 680
530 CALL HCHAR(X,Y,96)
540 IF Y>30 THEN 560
550 GOTO 370
560 PRINT :"YOU MADE IT!!"
570 FOR I=1 TO 15
580 CALL SOUND(100,INT(RND*500+500),4)
590 NEXT I
600 PRINT :"YOUR SCORE =";SC
610 IF BS<=SC THEN 630
620 BS=SC
630 PRINT :"BEST SCORE =";BS
640 GOTO 720
650 X=X+1
660 IF X<25 THEN 680
670 X=24
680 CALL SOUND(250,-5,4)
690 CALL HCHAR(X,Y,105)
700 PRINT :"YOU DID NOT MAKE IT ACROSS."
710 PRINT :"YOU MOVED";SC;"TIMES."
720 PRINT :::"TRY AGAIN? (Y/N)";
```

```
730 CALL KEY(0,K,S)
740 IF K=89 THEN 260
750 IF K<>78 THEN 730
760 CALL CLEAR
770 END
```

7

Astrostorm

Peter Lear
Translated for the TI by Patrick Parrish

Try to guide your spaceship, carrying emergency medical supplies, through a dangerous asteroid storm. A great deal depends on your skill as a navigator. The success or failure of your mission will often depend on your ability to make split-second decisions under pressure.

The mission: You are Captain Bosdiger of the interstellar tug *The Viccard*. While orbiting the fifth planet in the Benard system, you receive a distress call. The call comes from the Solarian system, in need of vital medical supplies. You are to pick them up from the sixth planet in the Benard system and then take them to the third planet in the Solarian system.

There is a time factor: the drugs have a short life. Your calculations indicate that it will be necessary to drop out of hyperspace between the fourth and fifth planets' orbits. There you will be in an asteroid field, which you must cross as quickly as possible.

Playing Astrostorm

The object of "Astrostorm" is to advance your spaceship across the asteroid field 12 times. Asteroids scroll from the bottom of the screen. Spaceship movement is horizontal. Control the movement of the ship by pressing the < and > keys.

The game can be quite challenging since there are several skill levels. As the game begins, the vertical position of the spaceship can be set nearer the bottom of the screen by specifying a greater "asteroid depth." A greater asteroid depth, of course, requires a faster reaction time. The difficulty of the game can be further increased by requesting a higher asteroid density (difficulty level).

The game loop (lines 500-830) is set to execute 1000 times. This means that you must finish your journey across the asteroid field before this loop is completed. You may find this time limit either too easy or too difficult, depending on the skill level you choose. If so, vary the limit in line 500 as you see fit.

To achieve a high score on Astrostorm, avoid moving your spaceship backwards since points are deducted from your total for each reverse move.

Astrostorm

```
100 RANDOMIZE
110 CALL CLEAR
120 FOR I=5 TO 8
130 CALL COLOR(I,16,1)
140 NEXT I
150 CALL CLEAR
160 PRINT "  A S T R O S T O R M !!!"
170 PRINT
180 PRINT
190 PRINT "POWER SHIP WITH < & > KEYS"
200 FOR I=1 TO 9
210 PRINT
220 NEXT I
230 FOR I=14 TO 3 STEP -1
240 FOR J=1 TO 20
250 NEXT J
260 CALL SCREEN(I)
270 NEXT I
280 FOR I=1 TO 400
290 NEXT I
300 S=3
310 PT=0
320 RSHIP=10
330 CSHIP=1
340 SHIP=62
350 COL=2
360 RLSHIP=RSHIP
370 CLSHIP=CSHIP
380 CALL CLEAR
390 PRINT "WHAT ASTEROID DEPTH (1-10)";
400 INPUT X
410 IF (X>10)+(X<1)THEN 380
420 PRINT
430 PRINT "WHAT DIFFICULTY LEVEL (1-10)"
440 INPUT DCULT
450 IF (DCULT>10)+(DCULT<1)THEN 380
460 CALL CLEAR
470 CALL SCREEN(S)
480 CALL COLOR(2,16,1)
490 REM  VARY LIMIT OF LOOP IN THE NEXT LINE
     IF THE GAME IS TOO HARD OR TOO EASY
500 FOR LOOP=1 TO 1000
510 FOR I=1 TO INT(RND*X)+1
520 COL=INT(RND*30)+2
530 CALL HCHAR(23,COL,42)
540 NEXT I
550 PRINT
```

```
560 CALL GCHAR(RSHIP+DCULT,CSHIP,LOC)
570 CALL HCHAR(RLSHIP+DCULT-1,CLSHIP,32)
580 IF LOC=42 THEN 1190
590 CALL SCREEN(S)
600 CALL HCHAR(RSHIP+DCULT,CSHIP,SHIP)
610 CLSHIP=CSHIP
620 RLSHIP=RSHIP
630 CALL KEY(3,A,STATUS)
640 IF A<>ASC(".")THEN 730
650 CALL SOUND(-700,-5,3)
660 PT=PT+5*DCULT*X
670 CSHIP=CSHIP+1
680 IF CSHIP<>32 THEN 720
690 CSHIP=1
700 S=S+1
710 IF S=15 THEN 1030
720 SHIP=62
730 IF A<>ASC(",")THEN 820
740 CALL SOUND(-700,-6,4)
750 SHIP=60
760 PT=PT-8*DCULT*X
770 CSHIP=CSHIP-1
780 IF CSHIP>0 THEN 810
790 CSHIP=32
800 S=S-SGN(S-3)
810 SHIP=60
820 REM
830 NEXT LOOP
840 CALL CLEAR
850 PRINT " YOU DID NOT COMPLETE YOUR"
860 PRINT
870 PRINT "MISSION. THE SOLARIAN"
880 PRINT
890 PRINT "GOVERNMENT HAS FINED YOU"
900 PRINT
910 PRINT PT;"GALACTIC CREDITS."
920 PRINT
930 PRINT
940 PRINT
950 PRINT "  BUT, THEY ALSO HAVE "
960 PRINT
970 PRINT "ANOTHER MISSION FOR YOU!!!"
980 PRINT
990 PRINT
1000 INPUT "ARE YOU GAME (Y OR N)?":G$
1010 IF G$="Y" THEN 300
1020 GOTO 1500
1030 REM  YOU WIN!!
1040 CALL CLEAR
```

```
1050 PRINT "  YOU COMPLETED YOUR "
1060 PRINT
1070 PRINT "MISSION. THE SOLARIAN MINI-"
1080 PRINT
1090 PRINT "STER HAS AWARDED YOU ";PT
1100 PRINT
1110 PRINT "SOLARIAN CREDITS FOR YOUR "
1120 PRINT
1130 PRINT "SERVICES AND WISHES YOU"
1140 PRINT
1150 PRINT "TO MAKE ANOTHER DELIVERY!"
1160 PRINT
1170 PRINT
1180 GOTO 980
1190 CALL SCREEN(12)
1200 FOR I=1 TO 50
1210 NEXT I
1220 CALL SCREEN(9)
1230 FOR VOL=24 TO 1 STEP 4
1240 CALL SOUND(200,-7,VOL)
1250 NEXT VOL
1260 FOR VOL=1 TO 24 STEP 4
1270 CALL SOUND(200,-7,VOL)
1280 NEXT VOL
1290 CALL SCREEN(12)
1300 FOR I=1 TO 10
1310 NEXT I
1320 CALL SCREEN(8)
1330 CALL CLEAR
1340 CALL SCREEN(6)
1350 PRINT "{4 SPACES}TOO BAD! THE SOLARIAN"
1360 PRINT " GOVERNMENT HAS SENT YOUR"
1370 PRINT " FAMILY ";PT;" GALACTIC"
1380 PRINT " CREDITS."
1390 FOR I=1 TO 5
1400 PRINT
1410 NEXT I
1420 PRINT "{3 SPACES}HIT -P- TO PLAY AGAIN"
1430 PRINT " OR -S- TO STOP."
1440 FOR I=1 TO 5
1450 PRINT
1460 NEXT I
1470 CALL KEY(3,K,ST)
1480 IF ST=0 THEN 1470
1490 IF (K=ASC("C"))+(K=ASC("P"))THEN 300
1500 END
```

8

Action Games

8

Goblin

Dan Goff
Translated for the TI by Patrick Parrish

Custom characters are used to create a simple entertaining game. The object is to capture the creatures with your goblin while avoiding the block-shaped obstacles that lie in your path.

Sad faces are positioned randomly among barriers on the screen. You need to capture the faces by landing on them. Use the left and right arrow keys to move your goblin. If you do not press a key, you will move straight upward. You will always be moving upward. When you reach the top of the screen, you wrap to the bottom of the screen. Try to clear the screen of all the faces. Notice that as you are capturing faces they frown at you, but if your goblin bumps into a barrier they all laugh with glee.

A running score is printed at the top left of the screen. The high score for previous games is printed as HS at the top center of the screen.

Program Explanation

CALL CHAR is used to define graphics characters or to redefine special characters. If there are already characters on the screen and you redefine a character, all graphics characters with that number will instantly change. So a CALL CHAR statement can change all the frowns to smiles at once.

Lines 400-460 place 50 random barriers on the screen, first making sure that there aren't already barriers in those random positions. If you want (or need) an easier game, reduce the number 50 in line 400.

Lines 470-600 place 27 sad faces in random positions, again making sure nothing else is already in those positions.

Z is the variable row number, and lines 650-670 keep you on the screen and return you to the bottom when you reach the top of the screen.

Wrapping takes place when you reach the top of the screen, returning you to the bottom.

Goblin

```
100 RANDOMIZE
110 GOTO 170
120 FOR I=1 TO LEN(H$)
130 R=ASC(SEG$(H$,I,1))
140 CALL HCHAR(ROW,XCOL+I,R)
150 NEXT I
160 RETURN
170 A=96
180 B=97
190 C=104
200 D=105
210 Z=24
220 COL=16
230 W=0
240 G=0
250 S=J
260 CALL CLEAR
270 GOSUB 1250
280 CALL SCREEN(16)
290 PRINT "{8 SPACES}G O B L I N"
300 PRINT
310 PRINT "{10 SPACES}HS : "
320 FOR I=1 TO 19
330 PRINT
340 NEXT I
350 PRINT "[=LEFT{14 SPACES}]=RIGHT";
360 ROW=4
370 XCOL=17
380 H$=STR$(HS)
390 GOSUB 120
400 FOR I=1 TO 50
410 X=INT(RND*30)+2
420 Y=INT(RND*16)+6
430 CALL GCHAR(Y,X,L)
440 IF L=B THEN 410
450 CALL HCHAR(Y,X,B)
460 NEXT I
470 FOR I=1 TO 27
480 X=INT(RND*30)+2
490 Y=INT(RND*16)+6
500 CALL GCHAR(Y,X,L)
510 IF (L=B)+(L=C)+(L=D)THEN 480
520 CALL GCHAR(Y+1,X-1,L)
530 CALL GCHAR(Y+1,X,M)
540 CALL GCHAR(Y+1,X+1,N)
550 IF (L<>B)+(M<>B)+(N<>B)THEN 590
560 CALL HCHAR(Y,X,D)
```

```
570 G=G+1
580 GOTO 600
590 CALL HCHAR(Y,X,C)
600 NEXT I
610 CALL SOUND(100,500,6)
620 CALL HCHAR(Z,COL,32)
630 IF L<>C THEN 650
640 CALL SOUND(10,880,4)
650 Z=Z-1
660 IF Z>4 THEN 680
670 Z=23
680 CALL KEY(0,L,ST)
690 IF (L<>83)*(L<>68)THEN 740
700 IF L<>83 THEN 730
710 COL=COL-SGN(COL-2)
720 GOTO 740
730 COL=COL+SGN(31-COL)
740 CALL GCHAR(Z,COL,L)
750 IF L=B THEN 1030
760 IF L=C THEN 820
770 CALL HCHAR(Z,COL,A)
780 FOR I=1 TO 25
790 NEXT I
800 IF W=27-G THEN 890
810 GOTO 620
820 W=W+1
830 S=S+25
840 H$=STR$(S)
850 ROW=4
860 XCOL=3
870 GOSUB 120
880 GOTO 770
890 J=S
900 CALL HCHAR(10,1,32,31)
910 GOSUB 120
920 H$="**** ALL RIGHT! ****"
930 XCOL=6
940 ROW=10
950 GOSUB 120
960 FOR I=1 TO 15
970 X=INT(RND*100)+300
980 CALL SOUND(75,X,8)
990 NEXT I
1000 FOR I=1 TO 100
1010 NEXT I
1020 GOTO 210
1030 REM WHOOPS! ...YOU CRASHED...
1040 CALL HCHAR(Z,COL,98)
1050 FOR I=3 TO 30 STEP 3
```

```
1060 CALL SOUND(50,-7,I)
1070 NEXT I
1080 CALL CHAR(104,"3C42A581A599423C")
1090 J=0
1100 IF S>HS THEN 1110 ELSE 1120
1110 HS=S
1120 H$="PLAY AGAIN (Y / N)?"
1130 ROW=22
1140 XCOL=2
1150 GOSUB 120
1160 CALL KEY(0,L,ST)
1170 IF ST=0 THEN 1160
1180 H$=CHR$(L)
1190 IF H$="Y" THEN 1230
1200 CALL CLEAR
1210 PRINT "SEE YA!"
1220 STOP
1230 CALL CHAR(104,"3C3CA58199A5423C")
1240 GOTO 210
1250 REM DEFINE CUSTOM CHARS
1260 REM CHAR 96 - GOBLIN
1270 CALL CHAR(96,"7EDBDBFFA55A5AA5")
1280 REM CHAR 97 - BARRIER
1290 CALL CHAR(97,"CCCC3333CCCC3333")
1300 REM CHAR 98 - CRUNCHED GOBLIN
1310 CALL CHAR(98,"CCCC33337EDBFFBD")
1320 REM CHAR - 104 - FROWN
1330 CALL CHAR(104,"3C3CA58199A5423C")
1340 REM CHAR - 105 - SMILE
1350 CALL CHAR(105,"3C42A581A599423C")
1360 CALL COLOR(10,7,1)
1370 CALL CHAR(91,"102040FF40201")
1380 CALL CHAR(93,"080402FF020408")
1390 FOR I=5 TO 8
1400 CALL COLOR(I,2,12)
1410 NEXT I
1420 RETURN
1430 END
```

8

Jumping Jack

Paul Burger
Translated for the TI by Charles Brannon

"Jumping Jack" demonstrates how sound can add excitement to a game.

Jack is running across platforms and climbing down ladders to get to the bottom of the screen. Sounds easy enough, right?

There's just one problem: these platforms are not very sturdy at all, and at any time they can collapse in certain places. You must be ready to press the space bar causing Jack to jump. If your timing is right, Jack will clear the hole and land safely on his feet. If not, Jack will fall into the collapsed section of the platform.

If you are not quick enough on the space bar, you still have a chance to clear the hole. Here's how: if the space bar is pressed immediately after Jack gets over the hole, you can make a saving jump. However, Jack must be over the hole while in the air to get points for jumping it, so no points are scored for using a saving jump to get over a hole.

A saving jump can also be used to jump two holes in a row. Simply make a saving jump as described above for the first hole, and Jack will fly over the second hole. This scores points only for the second hole, however.

If you guide Jack safely to the bottom of the screen, he must begin again at the top of a new screen with the ladders in different locations. As Jack goes through more screens, the platforms become less and less stable, so he'll have more holes to jump over.

Program Explanation

With only one moving character (made up of one graphics character), a game in TI BASIC can be fairly speedy. Jack automatically keeps moving, changing directions after each ladder.

Sounds can make any game more interesting. This program uses different noise numbers (negative numbers for frequency) in the CALL SOUND statement to create the sounds as Jack moves. Lines 680-700 use a very short duration with three musical tones for the jump. Lines 820-840 use a variable high-pitched frequency in a CALL SOUND loop for one effect, and lines 1040-1060 use a variable low-pitched frequency in the loop for a different effect.

Lines 370-430 randomly create the breaks in the platforms. The number of breaks depends on how many screens Jack has traveled through.

Jumping Jack

```
100 REM TI JUMPING JACK
110 RANDOMIZE
120 CALL CLEAR
130 GOSUB 1090
140 CALL CLEAR
150 DIFF=1
160 CALL SCREEN(16)
170 PRINT "LEVEL:";DIFF
180 DIR=1
190 PR=0
200 FOR I=2 TO 22 STEP 4
210 CALL HCHAR(I,1,96,32)
220 IF I>20 THEN 280
230 R=INT(RND*26+4)+DIR
240 IF (SGN(R-PR)<>DIR)THEN 230
250 CALL VCHAR(I,R,104,4)
260 PR=R
270 DIR=-DIR
280 NEXT I
290 COL=2
300 ROW=1
310 CHAR=112
320 OLDCOL=1
330 OLDROW=1
340 OLDCHAR=32
350 DIR=1
360 CALL HCHAR(OLDROW,OLDCOL,32)
370 IF RND>DIFF/10 THEN 440
380 R=INT(4*RND)*4+6
390 C=INT(RND*32)+1
400 CALL GCHAR(R,C,A)
410 IF A=104 THEN 440
420 CALL HCHAR(R,C,120)
430 CALL SOUND(100,-1,4)
440 CALL HCHAR(ROW,COL,CHAR-2*(DIR<0))
450 CALL SOUND(-5,-7,4)
460 IF ROW>20 THEN 1000
470 OLDCOL=COL
480 OLDROW=ROW
490 COL=COL+DIR
500 IF (COL>0)*(COL<33)THEN 550
510 COL=COL-DIR
```

```
520 ROW=ROW+4
530 DIR=-DIR
540 GOTO 360
550 CALL GCHAR(ROW+1,COL,CHECK)
560 CALL KEY(0,K,ST)
570 IF ST THEN 650
580 IF CHECK=120 THEN 780
590 IF CHECK<>104 THEN 620
600 DIR=-DIR
610 ROW=ROW+4
620 CHAR=225-CHAR
630 SCORE=SCORE+.5
640 GOTO 360
650 IF CHECK<>120 THEN 1040
660 CALL HCHAR(OLDROW,OLDCOL,32)
670 CALL HCHAR(ROW-1,COL,112-2*(DIR<0))
680 CALL SOUND(5,250,10)
690 CALL SOUND(5,200,10)
700 CALL SOUND(5,300,10)
710 CALL HCHAR(ROW-1,COL,128)
720 SCORE=SCORE+25
730 CALL SOUND(-500,500,1,510,10,520,20)
740 CALL SOUND(1,110,30)
750 CALL HCHAR(ROW-1,COL,32)
760 COL=COL+DIR
770 GOTO 500
780 CALL KEY(0,K,ST)
790 IF ST THEN 590
800 CALL HCHAR(OLDROW,OLDCOL,32)
810 CALL HCHAR(ROW,COL,116)
820 FOR I=1000 TO 1020
830 CALL SOUND(-1,I,0)
840 NEXT I
850 CALL HCHAR(ROW,COL,32)
860 CALL HCHAR(ROW+1,COL,121)
870 CALL SOUND(1000,-2,4,110,4)
880 CALL SOUND(1,110,1)
890 CALL CLEAR
900 CALL SCREEN(12)
910 PRINT "YOUR SCORE WAS:";INT(SCORE)
920 PRINT :"PLAY AGAIN? (Y/N):";
930 CALL KEY(0,K,ST)
940 IF (K<>ASC("Y"))*(K<>ASC("N"))THEN 930
950 PRINT CHR$(K)
960 IF K=ASC("N")THEN 990
970 SCORE=0
980 GOTO 140
990 STOP
1000 DIFF=DIFF+1
```

```
1010 SCORE=SCORE+50
1020 CALL CLEAR
1030 GOTO 170
1040 FOR I=150 TO 140 STEP -1
1050 CALL SOUND(-1,I,1)
1060 NEXT I
1070 SCORE=SCORE-25
1080 GOTO 590
1090 REM INITITIALIZE GAME, CHARACTERS
1100 FOR I=8 TO 19
1110 READ A,C
1120 CALL HCHAR(A,I,C)
1130 NEXT I
1140 DATA 13,74,12,85,11,77,10,80,11,73,12,7
     8,13,71,13,32,13,74,13,65,13,67,13,75
1150 PRINT "PRESS ANY KEY TO JUMP."
1160 READ A
1170 IF A=-1 THEN 1320
1180 READ A$
1190 CALL CHAR(A,A$)
1200 GOTO 1160
1210 DATA 96,FF422418182442FF
1220 DATA 104,7E427E427E427E42
1230 DATA 112,1028302478B82442
1240 DATA 113,102830A27C782448
1250 DATA 114,102818483C3A4884
1260 DATA 115,1028184A3C3C4824
1270 DATA 116,001C5D2A1C1C1422
1280 DATA 120,81814222242400C3
1290 DATA 121,BDBD5A22242400C3
1300 DATA 128,0077147741770000
1310 DATA -1
1320 FOR I=9 TO 13
1330 READ A
1340 CALL COLOR(I,A,1)
1350 NEXT I
1360 DATA 6,4,14,10,12
1370 PRINT : :"PRESS ANY KEY TO START.";
1380 CALL KEY(0,K,S)
1390 IF S<1 THEN 1380
1400 RETURN
1410 END
```

8

Air Defense

T. L. Wahl
Translated for the TI by Patrick Parrish

Play "Air Defense" and try to prevent the bombing of your city.

The object of "Air Defense" is to defend your city at the bottom of the screen from falling bombs. The bombs appear at various places at the top of the screen. As a bomb falls, line up the cross hairs of your gunsight with the bomb by using the arrow keys, then fire by pressing the space bar. If you prefer, you may use joysticks. You get only one shot, and timing is critical. The sooner you deactivate the bomb, the higher your score will be. After 20 bombs you will be shown your score with the number of hits and the number of misses.

Program Explanation

CALL SCREEN(*c*) will define the color *c* of the screen. A bomb explosion is shown by flashing screen colors (lines 1520-1660). Lines 1670-1720 create a bombing sound by varying the volume in a CALL SOUND loop.

If you have the TI-99/4 console, change all the lowercase letters in the instructions to regular capital letters.

If you use joysticks on the TI-99/4A console, be sure to release the ALPHA LOCK key.

If you have the TI Extended BASIC command module, you may wish to run this game in Extended BASIC for faster action. It will work as is, because no graphics characters are defined beyond color set 14. If you do use Extended BASIC, you may prefer to change the program by defining the bomb and the cross hairs as sprites.

Air Defense

```
100 DIM BUILDING(32,2)
110 RANDOMIZE
120 REM  BOMB CHARACTERS
130 CALL CHAR(129,"001CBEFFFFBE1C00")
140 CALL CHAR(37,"00000000001CBEFF")
150 CALL CHAR(38,"FFBE1C0000000000")
```

```
160 REM  CROSSHAIR CHARACTER
170 CALL CHAR(130,"181818FFFF181818")
180 CALL CLEAR
190 CALL SCREEN(12)
200 FOR J=9 TO 12
210 CALL COLOR(J,2,14)
220 NEXT J
230 T=0
240 P=0
250 Q=0
260 M=0
270 CALL CLEAR
280 PRINT "{8 SPACES}AIR DEFENSE"
290 PRINT
300 PRINT
310 PRINT "KEYBOARD OR JOYSTICK(k/j)?"
320 CALL KEY(0,KJ,ST)
330 IF ST=0 THEN 320
340 IF (KJ<>74)*(KJ<>75)THEN 320
350 KJ=KJ-73
360 FOR I=1 TO 4
370 PRINT
380 NEXT I
390 IF KJ=1 THEN 420
400 Z$="SPACE BAR."
410 GOTO 430
420 Z$="FIREBUTTON."
430 PRINT " do you need instructions?"
440 PRINT
450 PRINT "{8 SPACES}type Y or N"
460 FOR I=1 TO 4
470 PRINT
480 NEXT I
490 CALL KEY(0,Y,STATUS)
500 IF STATUS=0 THEN 490
510 IF Y=ASC("N")THEN 870
520 IF Y=ASC("Y")THEN 620
530 CALL CLEAR
540 PRINT
550 PRINT "  you did not press Y or N."
560 FOR I=1 TO 13
570 PRINT
580 NEXT I
590 FOR DELAY=1 TO 500
600 NEXT DELAY
610 GOTO 270
620 CALL CLEAR
630 PRINT "{3 SPACES}YOU MUST STOP THE FALLI
    NG"
```

```
640 PRINT "BOMB BY EXPLODING IT IN MID-AIR."
650 PRINT
660 PRINT
670 IF KJ=1 THEN 750
680 PRINT "{3 SPACES}-MOVE THE CROSSHAIR-"
690 PRINT
700 PRINT "  left :HOLD THE s KEY"
710 PRINT "  right:HOLD THE d KEY"
720 PRINT "  up{3 SPACES}:HOLD THE e KEY"
730 PRINT "  down :HOLD THE x KEY"
740 PRINT
750 PRINT "{3 SPACES}WHEN THE BOMB AND THE"
760 PRINT "CROSSHAIR ARE LINED UP,"
770 PRINT "FIRE BY PRESSING THE "
780 PRINT Z$
790 PRINT
800 PRINT "{3 SPACES}THE SOONER YOU GET THE"
810 PRINT "BOMB, THE HIGHER YOUR SCORE."
820 PRINT
830 PRINT
840 PRINT "{3 SPACES}PRESS any key TO START"
850 CALL KEY(0,S,STATUS)
860 IF STATUS=0 THEN 850
870 CALL CLEAR
880 PRINT "{7 SPACES}GOOD LUCK!!!"
890 FOR I=1 TO 10
900 PRINT
910 NEXT I
920 IF R=ASC("R")THEN 950
930 GOSUB 2040
940 GOTO 970
950 FOR I=1 TO 250
960 NEXT I
970 CALL CLEAR
980 GOSUB 2250
990 IF T=20 THEN 1820
1000 T=T+1
1010 CCROSS=16
1020 RCROSS=21
1030 RBOMB=1
1040 CALL SCREEN(6)
1050 CBOMB=INT(RND*29)+2
1060 H$=STR$(T)
1070 ROW=2
1080 COL=3
1090 GOSUB 2410
1100 SCORE=P*Q*10
1110 H$=STR$(SCORE)
1120 ROW=5
```

```
1130 GOSUB 2410
1140 FOR I=1 TO 70
1150 NEXT I
1160 FOR I=2 TO 5 STEP 3
1170 CALL HCHAR(I,3,32,6)
1180 NEXT I
1190 CALL VCHAR(RBOMB,CBOMB,129)
1200 RBOMB=RBOMB+1
1210 IF RBOMB=23 THEN 1500
1220 ON KJ GOSUB 2680,2480
1230 CALL VCHAR(RBOMB-1,CBOMB,37)
1240 CALL VCHAR(RBOMB,CBOMB,38)
1250 ON KJ GOSUB 2680,2480
1260 CALL VCHAR(RBOMB-1,CBOMB,32,2)
1270 GOTO 1190
1280 REM  BOMB DESTROYED
1290 RBOMB=RBOMB-1
1300 CALL SCREEN(10)
1310 CALL VCHAR(RBOMB,CBOMB,32,2)
1320 CNT=0
1330 C1=92
1340 C2=47
1350 FOR I=-1 TO 1 STEP 2
1360 CALL VCHAR(RBOMB+I,CBOMB+I,C1)
1370 CALL VCHAR(RBOMB+I,CBOMB-I,C2)
1380 NEXT I
1390 C1=32
1400 C2=32
1410 IF CNT=1 THEN 1470
1420 CNT=1
1430 FOR VOL=10 TO 30 STEP 5
1440 CALL SOUND(100,-6,VOL)
1450 NEXT VOL
1460 GOTO 1350
1470 P=P+1
1480 Q=Q+(23-RBOMB)
1490 GOTO 990
1500 REM  BOMB HITS THE CITY
1510 CALL VCHAR(22,CBOMB,32)
1520 CALL SCREEN(9)
1530 CALL COLOR(12,11,1)
1540 CALL VCHAR(23,CBOMB-1,122)
1550 CALL VCHAR(23,CBOMB,32)
1560 CALL VCHAR(23,CBOMB+1,123)
1570 CALL VCHAR(24,CBOMB-1,124)
1580 CALL VCHAR(24,CBOMB,125)
1590 CALL VCHAR(24,CBOMB+1,126)
1600 FOR I=1 TO 20
```

```
1610 NEXT I
1620 CALL COLOR(12,7,1)
1630 CALL SCREEN(12)
1640 FOR I=1 TO 20
1650 NEXT I
1660 CALL SCREEN(7)
1670 FOR VOL=24 TO 1 STEP 4
1680 CALL SOUND(200,-7,VOL)
1690 NEXT VOL
1700 FOR DVOL=1 TO 24 STEP 4
1710 CALL SOUND(200,-7,DVOL)
1720 NEXT DVOL
1730 FOR J=23 TO 24
1740 FOR I=CBOMB-1 TO CBOMB+1
1750 CALL VCHAR(J,I,32)
1760 NEXT I
1770 NEXT J
1780 CALL VCHAR(RCROSS,CCROSS,32)
1790 CALL COLOR(12,2,14)
1800 M=M+1
1810 GOTO 990
1820 CALL CLEAR
1830 CALL SCREEN(4)
1840 PRINT "{9 SPACES}GAME OVER"
1850 FOR I=1 TO 4
1860 PRINT
1870 NEXT I
1880 PRINT "{3 SPACES}DESTROYED{3 SPACES}";P
1890 PRINT
1900 PRINT "{3 SPACES}MISSED{6 SPACES}";M
1910 PRINT
1920 PRINT "{3 SPACES}TOTAL POINTS";P*Q*10
1930 FOR I=1 TO 4
1940 PRINT
1950 NEXT I
1960 PRINT "{3 SPACES}PRESS r TO PLAY AGAIN"
1970 PRINT
1980 PRINT
1990 CALL KEY(0,R,STATUS)
2000 IF STATUS=0 THEN 1990
2010 IF R=ASC("R")THEN 180
2020 END
2030 REM  READ CITY DATA
2040 FOR ROW=2 TO 1 STEP -1
2050 FOR COL=1 TO 32
2060 READ BUILDING(COL,ROW)
2070 NEXT COL
2080 NEXT ROW
2090 REM  CUSTOM CHAR & COLORS
```

```
2100 CALL CHAR(136,"FFABFFABFFABFFFF")
2110 CALL CHAR(128,"003C7EFFFFFF7E42")
2120 CALL CHAR(131,"42665A6642427E66")
2130 CALL CHAR(132,"6060606060606060")
2140 CALL CHAR(133,"607858F8D8F8D8F8")
2150 CALL CHAR(134,"F8A8F8A8F8A8F8F8")
2160 CALL CHAR(135,"C3C3FFABFFABFFFF")
2170 CALL COLOR(14,7,12)
2180 CALL CHAR(122,"8040201008040201")
2190 CALL CHAR(123,"0102040810204080")
2200 CALL CHAR(124,"80E0F8FEFFFFFFFF")
2210 CALL CHAR(125,"814224180081C3E7")
2220 CALL CHAR(126,"01071F7FFFFFFFFF")
2230 RETURN
2240 REM  SET UP CITY
2250 FOR ROW=2 TO 1 STEP -1
2260 FOR COL=1 TO 32
2270 CALL HCHAR(ROW+22,COL,BUILDING(COL,ROW)
     )
2280 NEXT COL
2290 NEXT ROW
2300 RETURN
2310 REM  CITY DATA
2320 DATA 136,134,131,135,133,136,136,133
2330 DATA 135,136,136,136,133,136,136,135
2340 DATA 135,136,136,134,133,136,136,136
2350 DATA 135,132,136,32,131,135,132,135
2360 DATA 134,133,128,32,132,32,135,32
2370 DATA 32,32,134,132,132,32,133,32
2380 DATA 32,32,128,32,132,32,133,135
2390 DATA 32,132,132,32,128,32,132,32
2400 REM  HORIZONTAL # PRINTER{4 SPACES}
2410 FOR I=1 TO LEN(H$)
2420 DIG=ASC(SEG$(H$,I,1))
2430 CALL HCHAR(ROW,COL+I,DIG)
2440 NEXT I
2450 RETURN
2460 REM  CROSSHAIR SUB
2470 REM  KEYBOARD ROUTINE
2480 OLDRCROSS=RCROSS
2490 OLDCCROSS=CCROSS
2500 CALL KEY(0,K,S)
2510 IF K<>69 THEN 2530
2520 RCROSS=RCROSS-SGN(RCROSS-1)
2530 IF K<>88 THEN 2550
2540 RCROSS=RCROSS+SGN(22-RCROSS)
2550 IF K<>68 THEN 2570
2560 CCROSS=CCROSS+SGN(32-CCROSS)
2570 IF K<>83 THEN 2590
```

```
2580 CCROSS=CCROSS-SGN(CCROSS-2)
2590 IF S=0 THEN 2610
2600 CALL VCHAR(OLDRCROSS,OLDCCROSS,32)
2610 CALL VCHAR(RCROSS,CCROSS,130)
2620 IF (RCROSS=RBOMB)+(RCROSS=RBOMB-1)THEN
     2630 ELSE 2660
2630 IF CCROSS<>CBOMB THEN 2660
2640 CALL KEY(0,K,S)
2650 IF K=32 THEN 1290
2660 RETURN
2670 REM  JOYSTICK ROUTINE - RELEASE ALPHA-L
     OCK KEY
2680 OLDRCROSS=RCROSS
2690 OLDCCROSS=CCROSS
2700 CALL JOYST(1,XRET,YRET)
2710 CCROSS=CCROSS+XRET/4
2720 IF CCROSS>32 THEN 2740
2730 IF CCROSS<1 THEN 2760 ELSE 2770
2740 CCROSS=32
2750 GOTO 2770
2760 CCROSS=1
2770 RCROSS=RCROSS-YRET/4
2780 IF RCROSS>22 THEN 2800
2790 IF RCROSS<1 THEN 2820 ELSE 2830
2800 RCROSS=22
2810 GOTO 2830
2820 RCROSS=1
2830 IF (RCROSS=OLDRCROSS)*(CCROSS=OLDCCROSS
     )THEN 2850
2840 CALL VCHAR(OLDRCROSS,OLDCCROSS,32)
2850 CALL VCHAR(RCROSS,CCROSS,130)
2860 IF (RCROSS=RBOMB)+(RCROSS=RBOMB-1)THEN
     2870 ELSE 2900
2870 IF CCROSS<>CBOMB THEN 2900
2880 CALL KEY(1,K,S)
2890 IF K=18 THEN 1290
2900 RETURN
2910 END
```

9

Extended BASIC Games

9

Games in TI Extended BASIC

For even more programming capabilities and for faster action and smoothly moving sprites, you may use TI Extended BASIC. Extended BASIC is a programming language on a command module. No other peripherals are required. The module adds 36K of preprogrammed memory. A manual and a reference card are included with the command module to help you with your programming.

Extended BASIC allows multiple-statement lines, which can make your programs more efficient, and programs usually execute faster than with standard console BASIC. Extended BASIC allows commands after IF-THEN-ELSE conditions, and you can use complex IF-THEN logic—very helpful in game conditions. The CALL MAGNIFY command allows different magnification factors, so it is possible to create a rather large graphics character with relatively little character definition. Extended BASIC also allows speech capabilities if you have the TI Speech Synthesizer peripheral.

One of the main uses of Extended BASIC is for action or arcade-style games. Extended BASIC allows the use of up to 28 sprites—smoothly moving graphic shapes that can pass over other characters without disturbing the screen display. First define your shape in high-resolution graphics if you wish (and you may define up to four shapes with one command). Then, using one CALL SPRITE statement, you specify the sprite number, the character number for the graphics definition, the color you desire, the starting location, and the velocity. Actually, you can define several sprites with one statement if you want.

Several subprograms that relate to sprites are provided in Extended BASIC. The games in this section describe these commands as they are used in the games.

9

Balloons

C. Regena

This short, simple game uses TI Extended BASIC. TI Extended BASIC command module required.

Seven balloons have been released and rise upward. Your objective is to pop them as quickly as possible. Use the arrow keys to move your white spear to poke each balloon.

Program Explanation

The purpose of this very short program is to show how Extended BASIC can work. Lines 120-130 print the instructions on the screen. DISPLAY AT is used so the printing will not scroll. The double colon separates commands for multistatement lines. Line 140 defines graphic characters. The first command actually defines both character 96 and character 97—you can define up to four characters with one CALL CHAR. The top score, TS, or best time is initialized to 9999.

Lines 150-160 give the player time to read the instructions. The computer waits until the player presses ENTER. Notice in line 160 that the line number after THEN in the IF-THEN statement can refer to the same line.

Line 170 clears the screen and prints "TIME:" in the upper-right corner. The variables T, B, X, and Y are all initialized to zero in one statement. T is time, B is the number of balloons popped, and X and Y are the row and column velocities of the spear.

Lines 180-200 draw the sprites for the game. The variable I is the sprite number, which is also used in calculating the beginning column coordinate where the sprite starts. Line 190 chooses a random color (not cyan) for each balloon. The first statement in line 200 defines the balloon sprite—sprite number I, shape of character 96, color R which was chosen in line 190—starting in dot-row 185, which is at the bottom of the screen, and dot-column I*8+1, with upward random row velocity from 1 to 6. Seven balloons are drawn. Sprite #1 is the spear, drawn at the right side of the screen at a random row. The CALL SOUND statement sounds a beginning "beep."

Line 210 detects if a key is pressed. The time is T and is incre-

mented within each CALL KEY loop to simulate realtime. DISPLAY AT can print the time without scrolling, so graphics are not affected. CALL COINC(ALL,C) detects if there is "coincidence" or a collision among ALL (any) of the sprites. If so, C is −1. If there is no coincidence, C is zero. Line 220 returns to the CALL KEY command if an arrow key was not pressed.

Lines 230-270 define the dot-row and dot-column velocity factors depending on the arrow key pressed. CALL MOTION in line 280 moves Sprite #1 at velocity X and Y. Coincidence is checked again.

Lines 290-320 contain the procedure if coincidence is detected. There is a balloon-popping sound, then CALL DISTANCE checks the distance between two sprites at a time—#1 and each of the other sprites—to determine which balloon popped. If the distance D is less than 425 (an arbitrary number chosen after experimentation), then that particular sprite number is the balloon that popped. Line 320 changes the shape of the balloon with CALL PATTERN then erases the balloon with CALL DELSPRITE. If there are still more balloons, the program branches back to line 210.

Line 330 clears all sprites then determines the lowest time. An IF-THEN statement is not limited to line numbers after THEN or ELSE, but can contain a command such as TS = T.

Line 340 displays the fastest time, TS, then plays an ending random tune. Lines 350-370 present the option to try again, and the program branches according to the player's response. Line 380 clears the screen and ends the program.

Variations

CALL DISTANCE detects which balloon is popped by the spear. The D value of 425 may not be close enough by the time the later sprites are checked in the loop, so the wrong balloon could disappear. You could number the balloons and have the player pop the balloons in order. In this case, instead of CALL COINC(ALL,C) you would check coincidence between two specific sprites, #1 and the particular balloon number. You might then try adding a random column velocity to each balloon so they all don't go straight upward.

Balloons

```
100 REM TI EXTENDED BASIC
110 CALL CLEAR
```

```
120 DISPLAY AT(4,4):"** B A L L O O N S **"
    :: DISPLAY AT(7,4):"USE THE ARROW KEYS T
    O" :: DISPLAY AT(9,4):"MOVE YOUR WHITE S
    PEAR."
130 DISPLAY AT(12,4):"TRY TO POP ALL THE" ::
     DISPLAY AT(14,4):"BALLOONS IN AS LITTLE
    " :: DISPLAY AT(16,4):"TIME AS POSSIBLE.
    "
140 CALL CHAR(96,"3C7EFFFFFFFF7E3C8142183C3C
    184281"):: CALL CHAR(104,"101038FE38101"
    ):: TS=9999
150 DISPLAY AT(23,4):"PRESS <ENTER> TO START
    ."
160 CALL KEY(0,K,S):: IF K<>13 THEN 160
170 CALL CLEAR :: DISPLAY AT(1,1):"TIME:" ::
     T,B,X,Y=0
180 FOR I=2 TO 26 STEP 4
190 RANDOMIZE :: R=INT(RND*12)+3 :: IF R=8 T
    HEN 190
200 CALL SPRITE(#I,96,R,185,I*8+1,-INT(RND*6
    +1),0):: NEXT I :: CALL SPRITE(#1,104,16
    ,INT(RND*180+1),233):: CALL SOUND(150,14
    97,2)
210 CALL KEY(1,K,S):: T=T+1 :: DISPLAY AT(1,
    7):T :: CALL COINC(ALL,C):: IF C=-1 THEN
     290
220 IF (K+1<1)+(K>5)THEN 210
230 ON K+1 GOTO 250,210,260,270,210,240
240 X=-5 :: Y=0 :: GOTO 280
250 X=5 :: Y=0 :: GOTO 280
260 X=0 :: Y=-5 :: GOTO 280
270 X=0 :: Y=5
280 CALL MOTION(#1,X,Y):: CALL COINC(ALL,C):
    : IF C=0 THEN 210
290 CALL SOUND(50,-5,0)
300 FOR I=26 TO 2 STEP -4 :: CALL DISTANCE(#
    1,#I,D):: IF D<425 THEN 320
310 NEXT I :: I=2
320 CALL PATTERN(#I,97):: CALL DELSPRITE(#I)
    :: B=B+1 :: IF B<7 THEN 210
330 CALL DELSPRITE(ALL):: IF T<TS THEN TS=T
340 DISPLAY AT(18,4):"FASTEST TIME =";TS ::
    FOR B=1 TO 15 :: CALL SOUND(100,INT(RND*
    1000)+500,2):: NEXT B
350 DISPLAY AT(22,4):"TRY AGAIN? (Y/N)"
360 CALL KEY(0,K,S):: IF K=89 THEN 170
370 IF K<>78 THEN 360
380 CALL CLEAR :: END
```

9

Diamond Drop

Matt Giwer
Translated for the TI by Patrick Parrish

Relying on the outstanding sprite capability of TI Extended BASIC, "Diamond Drop" offers a game with quick, smooth action.

A series of vertically positioned paddles help you catch colorful diamonds raining from the top of the screen in "Diamond Drop." These paddles can be controlled with the keyboard or with joystick 1. With keyboard control, the S and D keys are presently used for left and right movement. If you are more comfortable using some other keys, however, simply substitute the ASCII values corresponding to the desired keys in lines 420 and 430 for the numbers 68 and 83. (To find the ASCII value of a key, use PRINT ASC("X"), where X is the key you want to use.)

Skill Levels

Two overall skill levels, determined by how fast the diamonds drop, are offered in the game. After you clear the entire screen of diamonds, the drop speed is increased. For the first screen, drop speed is 25 for skill level one, and 40 for skill level two. This is set in line 250. Line 560 increases the drop speed by three with completion of each screen.

To make the game more challenging, the diamonds can be dropped at a random diagonal angle. With this feature, some interesting playing situations will develop. Since screen wrap-around of the paddles is permitted, you must often make quick decisions about which direction to move. A wrong move will ultimately affect your score since only ten misses are allowed.

Scoring in the game, as determined in line 510, is affected by a number of factors. First, more points are awarded for diamonds garnered from successively higher rows on the screen. Second, diamond values increase with completion of each screen. Third, points are accumulated twice as quickly at skill level two. And last, if you choose to add an angle of descent to each diamond, a greater number of points are given based on the severity of the descent angle. When ten diamonds have been missed, the game

is over, and your score and the high score for the session are posted.

Extended BASIC for the TI-99/4A features some convenient commands for sprite manipulation. Since sprite movement can be very fast, detection of collisions between sprites is not infallible. As noted in the *TI Extended BASIC Manual*, sprites which coincide in position can be detected only when the COINC subprogram is CALLed from BASIC. Thus, if your program is executing some statement other than CALL COINC when sprites cross, no collision will be detected. Fortunately, this causes a problem only at the most advanced levels of the game.

Diamond Drop

```
10 REM TI EXTENDED BASIC
100 DIM KOLOR(6)
110 RANDOMIZE
120 GOSUB 630
130 REM 108-DEFINE DIAMOND SPRITE CHAR,128-1
    36 ARE THE PADDLES
140 CALL CHAR(108,"10387CFE7C381000000000000
    0000000000000000000000000000000000000000"
    )
150 CALL CHAR(128,"FFFFFFFF0000FFFFFFFF0000F
    FFFFFFFFFFFFFFF0000FFFFFFFF0000FFFFFFFF"
    )
160 CALL CHAR(132,"000000000000FFFFFFFF0000F
    FFFFFFF000000000000FFFFFFFF0000FFFFFFFF"
    )
170 SCR=0 :: SK=0 :: CH=10 :: S=0 :: CALL CL
    EAR :: CALL SCREEN(16):: DISPLAY AT(4,9)
    :"D I A M O N D"
180 FOR ROW=3 TO 6
190 CALL HCHAR(ROW+2,6,32,20)
200 DISPLAY AT(ROW+3,6):"```  ```{3 SPACES}`
    `  ```" :: DISPLAY AT(ROW+4,6):"h  h h
    h h  h h  h"
210 DISPLAY AT(ROW+5,6):"p  p p  p p  p p  p
    " :: DISPLAY AT(ROW+6,6):"x  x xxx  x  x
     xxx"
220 DISPLAY AT(ROW+7,6):"h  h h  h h  h h" :
    : DISPLAY AT(ROW+8,6):"```  `  `  ``  `"
230 NEXT ROW
240 DISPLAY AT(18,4):"SKILL LEVEL (1,2) ?" :
    : ACCEPT AT(18,24)BEEP VALIDATE("12")SIZ
    E(1):SK$ :: SK=VAL(SK$)
250 DROP=25 :: IF SK=2 THEN DROP=40 :: REM C
    HANGE DROP RATE TO CHANGE DIFFICULTY
```

```
260 DISPLAY AT(21,2):"DROP WITH ANGLE (Y/N)
    ?" :: ACCEPT AT(21,26)BEEP VALIDATE("YN"
    )SIZE(1):ANG$
270 IF ANG$="N" THEN ANG=0 :: GOTO 290
280 ANG=1
290 CALL CLEAR :: SCR=SCR+1
300 DISPLAY AT(1,2):"CHANCES:";CH :: DISPLAY
     AT(1,15):"SCORE:";S
310 ROW=3 :: FOR I=96 TO 120 STEP 8
320 CALL HCHAR(ROW,3,I,28):: ROW=ROW+1 :: NE
    XT I
330 CALL HCHAR(24,1,30,32)
340 CALL MAGNIFY(4):: CALL SPRITE(#1,128,5,1
    50,115,0,H)
350 KHAR=108 :: ROW=41 :: FOR J=6 TO 3 STEP -1
360 A$="" :: FOR I=3 TO 30 :: A$=A$&CHR$(I):
    : NEXT I :: N=28
370 IF N=0 THEN 530
380 R=INT(LEN(A$)*RND+1):: P$=SEG$(A$,R,1)::
     X=ASC(P$):: N=N-1 :: IF N=0 THEN 400
390 A$=SEG$(A$,1,R-1)&SEG$(A$,R+1,LEN(A$)-R)
400 B=INT(RND*61*ANG)-30*ANG
410 CALL HCHAR(J,X,32):: CALL SPRITE(#2,KHAR
    ,KOLOR(J),ROW,8*(X-1)-2,DROP,B)
420 CALL KEY(0,K,ST):: IF ST=0 THEN CALL JOY
    ST(1,H,V):: H=SGN(H)ELSE H=(K=83)-(K=68)
430 CALL MOTION(#1,0,60*H)
450 CALL COINC(ALL,C):: IF C THEN 510
460 CALL POSITION(#2,DROW,DCOL):: IF DROW<15
    5 THEN 420
470 CALL POSITION(#1,PROW,PCOL):: IF (DCOL-P
    COL<16)*(DCOL-PCOL>-8)THEN 510
480 CALL DELSPRITE(#2):: CALL MOTION(#1,0,0)
    :: CH=CH-1 :: CALL SCREEN(11):: FOR F=0
    TO 25 STEP 5
490 CALL SOUND(-200,-5,F):: NEXT F :: CALL S
    CREEN(16):: IF CH=0 THEN GOTO 570
500 GOTO 520
510 CALL DELSPRITE(#2):: CALL MOTION(#1,0,0)
    :: S=S+(60/J)*SK*SCR+(60/J)*SK*SCR*INT(A
    BS(B)/15)
520 DISPLAY AT(1,2):"CHANCES:";CH :: DISPLAY
     AT(1,15):"SCORE:";S :: GOTO 370
530 K=K+4 :: ROW=ROW-8 :: M=128 :: IF J<6 TH
    EN M=132
540 FOR F=0 TO 30 STEP 6 :: CALL SOUND(-300,
    1500,F):: NEXT F
550 CALL SPRITE(#1,M,5,150,115,0,H)
```

```
560 NEXT J :: FOR G=600 TO 1400 STEP 100 ::
    CALL SOUND(100,G,1):: NEXT G :: DROP=DRO
    P+3 :: GOTO 290
570 CALL SCREEN(14):: IF S>HS THEN HS=S
580 CALL DELSPRITE(ALL):: CALL CLEAR :: DISP
    LAY AT(8,5):"YOUR SCORE: ";S :: DISPLAY
    AT(11,5):"HIGH SCORE: ";HS
590 DISPLAY AT(16,5):"PLAY AGAIN (Y/N)? " ::
     ACCEPT AT(16,24)BEEP VALIDATE("NY")SIZE
    (1):REPLY$
600 IF REPLY$="N" THEN 620
610 GOTO 170
620 STOP
630 REM DEFINE SMALL DIAMONDS AND COLORS
640 FOR I=96 TO 120 STEP 8
650 CALL CHAR(I,"10387CFE7C381000"):: NEXT I
660 CALL COLOR(11,11,1)
670 CALL COLOR(9,3,1)
680 CALL COLOR(10,10,1)
690 CALL COLOR(12,14,1)
700 FOR J=3 TO 6 :: READ KOLOR(J):: NEXT J
710 DATA 3,10,11,14
720 RETURN
```

9

Trapshoot

C. Regena

Besides being a fast-action shoot game, "Trapshoot" is an excellent example of how to use sprites in your own programs. TI Extended BASIC module required.

It's time to try your skill at the TI trapshooting range. Each game consists of 50 clay pigeons sprung from the trap near the center of the screen. You aim the shotgun located at the bottom center of the screen by pressing the left or right arrow key. Fire when you are ready by pressing the ENTER key. You'll need to aim and fire quickly—you have only one chance per target.

At the lower-right side of the screen is your scoring record. The number of successful hits, the number of clay pigeons, and the number of rounds fired are displayed.

Programming Techniques

Line 120 defines a function RRV for the random row velocity for the clay pigeon moving from 0 to 140 upward. You may change the number 14 in the equation to 15 or 16 to make the target move upward more quickly, but you will have less time to aim the shotgun, shoot, and hit the target.

Line 130 defines a function RCV for the random column velocity of −17 to +17, moving the target toward the left or right. The number 18 in the equation may be changed to decrease or increase the range of the target. Increasing the number will move the target more to the left or right, but the target may wrap to the other side of the screen before being deleted.

The shotgun is Sprite #3, defined in line 190. There are seven shotgun positions drawn with characters 108 through 135. CALL MAGNIFY(4) is used so the shotgun may be drawn as large as possible and still use only one character number for the sprite. If the left arrow key is pressed, the character number N is decreased by 4; if the right arrow key is pressed, N is increased by 4. N may vary from 108 to 132, where 120 is straight up. The shotgun position is changed after a key is pressed by using CALL PATTERN (#3,N).

Line 220 stops the game after 50 clay pigeons. You may

change the limits of the game by changing the limit for T, or you may wish to test for the number of shots, SH (perhaps stopping after 50 rounds or 100 rounds instead of after 50 birds).

Line 230 springs the clay pigeon from the trap at the random row and column velocities. The target is Sprite #1.

If you press ENTER to fire, the shot appears as Sprite #2 at the end of the shotgun and goes upward in the direction the shotgun is aimed. The value of N2 is N-120 and is used in calculating position and column velocity parameters for Sprite #2. The position of the end of the shotgun is dot-column 116 plus some function of N2. Experimentation shows that the dot-column position is 116 + N2*1.2.

Using trigonometry, the angle of the shotgun is determined dependent upon the character number. The ratio of the row velocity to the column velocity is equal to the ratio of the horizontal displacement to the vertical length of the shotgun. Whether the shotgun is pointing left or right is determined by SGN(N2). The upward (row) velocity of Sprite #2 was set at 100.

The theoretical factor to calculate column velocity is 12.5, but since the displacement per character number is not precisely linear, 12.7 works better. The resultant column velocity is (N2/4 + 2*SGN(N2))*12.7. The row velocity of 100 was chosen so the shot moves faster than the clay pigeon, but slowly enough to report coincidence and to prevent wrapping on the screen.

Controlling Sprites

Lines 330-340 check to see if the shot hits the target. CALL COINC(ALL,C) is used so coincidence is reported if any dot of the shot coincides with any dot of the target. Using a statement such as CALL COINC(#1,#2, TOL,C) between two sprites tests coincidence of the upper-left corners of each sprite within a certain tolerance; sometimes a hit would be scored when the shot appeared to miss the target.

The faster sprites move, the more difficult it is to control them in a program. Coincidence is reported only if the sprites are touching at the exact moment the CALL COINC statement is executed in the program. Once ENTER is pressed and the shot starts on its path, CALL COINC is executed in a FOR-NEXT loop 19 times. At the end of 19 loops without coincidence, the shot is near the top of the screen and is deleted.

If coincidence is reported, then the program branches to the appropriate section for a hit. If you change the speed of either the

target or the shot, you may need to change the limit 19 in the FOR-NEXT loop. If you play many times, you may notice that occasionally the shot will pass through the target without recording a hit. This happens when the target is going straight upward slowly and you fire immediately. The sprites pass each other before the program has a chance to get to the CALL COINC statement. To avoid this problem, you could slow the shot down. However, I prefer the faster shot since the problem rarely occurs. This is an example of "programming trade-offs."

After the shot is deleted, the program keeps testing the position of the target until it is at the top of the screen; then Sprite #1, the target, is deleted (line 360). If ENTER is not pressed, then the position of Sprite #1 is tested in the CALL KEY loop.

If the target is hit, then the broken clay pigeon is shown by changing the pattern of the sprite.

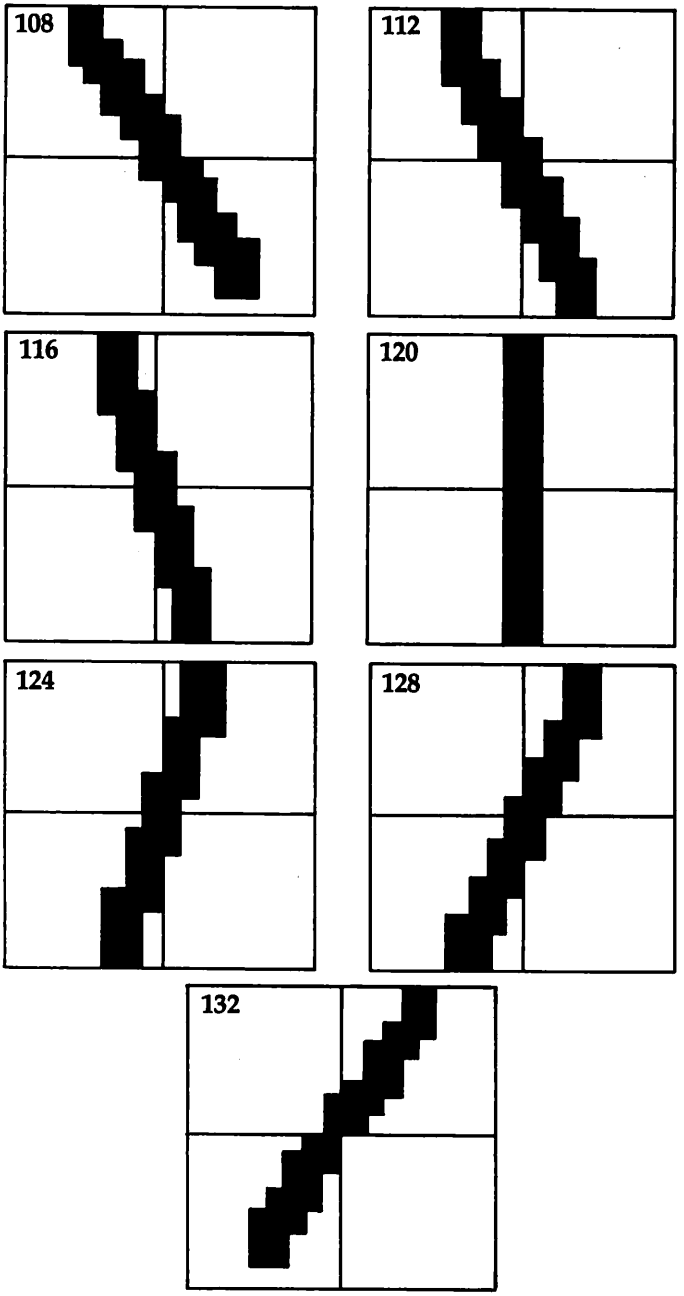
The shot disappears when the pattern of the shot is changed to a blank character. The statement is CALL PATTERN (#1,100,#2, 136). After sounding a hit using Noise −6, both sprites are deleted with CALL DELSPRITE(#1,#2).

99/4 Versus 99/4A

Some of the consoles process at different rates. It makes a difference whether you have the TI-99/4, the earlier TI-99/4A, or the later TI-99/4A. It also makes a difference if you have the old Extended BASIC module or the new Extended BASIC. (You can tell which you have by holding a key down. If it will automatically repeat, you have a newer module.) Since this game is very critical on timing, you will have to experiment a little so that sprites won't wrap and cause bugs. You can adjust the game by changing the limit in line 330.

New XBASIC, TI-99/4A	**330 FOR I = 1 TO 19**	**(or 20)**
Old XBASIC, TI-99/4A	**330 FOR I = 1 TO 9**	
New XBASIC, TI-99/4	**330 FOR I = 1 TO 19**	
Old XBASIC, TI-99/4	**330 FOR I = 1 TO 12**	

Graphic Representation of the Shotgun Positions



Trapshoot

```
100 REM TRAPSHOOT
110 REM TI EXTENDED BASIC
120 DEF RRV=-INT(RND*10+14)
130 DEF RCV=(-1)^(INT(RND*4+1))*(INT(RND*18)
    )
140 GOTO 460
150 RANDOMIZE :: CALL CLEAR :: N=120 :: H,T,
    SH=0
160 CALL COLOR(8,3,1):: CALL HCHAR(24,1,92,3
    2)
170 CALL HCHAR(15,14,140):: CALL HCHAR(15,15
    ,141,3):: CALL HCHAR(15,18,142)
180 CALL HCHAR(16,14,141,5):: CALL HCHAR(17,
    14,141,5)
190 CALL SPRITE(#3,N,5,160,108)
200 DISPLAY AT(21,19):"HITS:" :: DISPLAY AT(
    22,19):"BIRDS:" :: DISPLAY AT(23,19):"RO
    UNDS:"
210 DISPLAY AT(21,26):USING "###":H :: DISPL
    AY AT(22,26):USING "###":T :: DISPLAY AT
    (23,26):USING "###":SH
220 IF T=50 THEN 410
230 T=T+1 :: CALL SPRITE(#1,96,7,112,117,RRV
    ,RCV):: CALL SOUND(150,-5,0)
240 CALL KEY(0,KEY,S)
250 IF KEY=13 THEN 310
260 IF KEY<>83 THEN 280 ELSE N=N-4 :: IF N<1
    08 THEN N=108
270 CALL PATTERN(#3,N):: GOTO 300
280 IF KEY<>68 THEN 300 ELSE N=N+4 :: IF N>1
    32 THEN N=132
290 CALL PATTERN(#3,N)
300 CALL POSITION(#1,R,C):: IF R>10 AND R<11
    2 THEN 240 ELSE CALL DELSPRITE(#1):: GOT
    O 210
310 CALL SOUND(1000,-4,0):: N2=N-120
320 CALL SPRITE(#2,104,2,154,116+N2*1.2,-100
    ,(N2/4+2*SGN(N2))*12.7)
330 FOR I=1 TO 19 :: CALL COINC(ALL,C):: IF
    C=-1 THEN 370
340 NEXT I
350 CALL DELSPRITE(#2)
360 CALL POSITION(#1,R,C):: IF R>5 AND R<112
     THEN 360 ELSE CALL DELSPRITE(#1):: SH=S
    H+1 :: GOTO 210
370 CALL PATTERN(#1,100,#2,136):: CALL SOUND
    (1000,-6,0)
```

```
380 CALL DELSPRITE(#1,#2)
390 SH=SH+1 :: H=H+1
400 CALL SOUND(1,-6,30):: GOTO 210
410 CALL DELSPRITE(ALL):: CALL HCHAR(24,1,32
    ,32):: CALL COLOR(8,2,1)
420 PRINT : :TAB(4);"SCORE =";INT(H*100/T+.5
    );"PERCENT": :TAB(4);"TRY AGAIN? (Y/N)"
430 CALL KEY(0,KEY,S)
440 IF KEY=89 THEN 150
450 IF KEY=78 THEN STOP ELSE 430
460 CALL CLEAR :: CALL MAGNIFY(4)
470 CALL CHAR(96,"3C7EFFFFFFFF7E3C0000000000
    0000000000000000000000000000000000000000")
    :: CALL COLOR(2,7,16)
480 CALL HCHAR(9,6,42,21):: CALL HCHAR(13,6,
    42,21)
490 CALL VCHAR(10,6,42,3):: CALL VCHAR(10,26
    ,42,3)
500 DISPLAY AT(11,6)SIZE(17):"T R A P S H O
    O T"
510 CALL CHAR(64,"3C4299A1A199423C")
520 REM
530 CALL CHAR(104,"10381000000000000000000000
    0000000000000000000000000"):: CALL COLOR
    (2,16,7)
540 CALL CHAR(108,"18181C0E06070301010000000
    0000000000000000008080C0E06070381818"):
    : CALL COLOR(2,7,16)
550 CALL CHAR(112,"000C0C0E06070303010100000
    0000000000000000000008080C0C0E060703030"
    ):: CALL COLOR(2,16,7)
560 CALL CHAR(116,"0606060703030301010100000
    0000000000000000000808080C0C0C0E0606060"
    ):: CALL COLOR(2,7,16)
570 CALL CHAR(120,RPT$("01",16)&RPT$("80",16
    )):: CALL COLOR(2,16,7)
580 CALL CHAR(124,"0000000000000101010303030
    7060606606060E0C0C0C080808"):: CALL COLO
    R(2,7,16)
590 CALL CHAR(128,"0000000000000001010303070
    60E0C0C0030307060E0C0C0808"):: CALL COLO
    R(2,16,7)
600 CALL CHAR(132,"0000000000000101030706 0E1
    C1818001818387060E0C08080"):: CALL COLOR
    (2,7,16)
610 CALL COLOR(9,11,1):: CALL COLOR(2,16,7)
620 CALL CHAR(140,"0103070F1F3F7FFFFFFFFFFFF
    FFFFFFF80C0E0F0F8FCFEFF0"):: CALL COLOR(
    2,7,16)
```

```
630 CALL CHAR(100,"DCDD590244D29B19000000000
    00000000000000000000000000000"):: CALL COLO
    R(2,16,7)
640 CALL COLOR(14,11,1):: CALL COLOR(2,7,16)
650 CALL CHAR(136,"0"):: CALL COLOR(2,16,7)
660 CALL CHAR(92,"82A6B7F7FFFFFFFF"):: CALL
    COLOR(2,7,16)
670 CALL COLOR(2,2,1):: CALL CLEAR
680 DISPLAY AT(3,1):"A CLAY PIGEON WILL SPRI
    NG{5 SPACES}":"FROM THE TRAP."
690 DISPLAY AT(8,1):"USE THE ARROW KEYS TO A
    IM{4 SPACES}":"YOUR RIFLE LEFT OR RIGHT.
    "
700 DISPLAY AT(13,1):"PRESS <ENTER> TO SHOOT
    ."
710 DISPLAY AT(17,1):"YOU WILL HAVE A CHANCE
    {7 SPACES}":"TO SHOOT 50 CLAY BIRDS."
720 DISPLAY AT(24,1):"PRESS ANY KEY TO START
    ."
730 CALL KEY(0,KEY,S):: IF S=1 THEN 150 ELSE
     730
740 END
```

9

Rocket Duel

C. Regena

This challenging two-player game requires the TI Extended BASIC command module.

This Extended BASIC game is designed for two players. The player at the left controls the red rocket with the standard arrow keys on E, S, D, X, and shoots with the F key. The player at the right controls the blue rocket with the I, J, K, M keys and shoots with the H key. Each player must shoot the other player's rocket without getting hit and without colliding.

Program Explanation

CALL MAGNIFY(3) is used so each sprite is four characters and enlarged. Lines 140-230 define the graphics characters four at a time. Each rocket is drawn in eight positions, up, down, left, right, and the four diagonals. The variables A1 and A2 keep track of the attitudes, or the positions the rockets are pointing.

The players choose from three skill levels. Velocities of the sprites will depend on the level chosen. Line 280 defines Sprites #1 and #2 as the two rockets. Line 290 uses the split keyboard to detect which arrow keys the players press. Depending on the arrow key (K1 and K2) pressed, the direction of the rocket is checked, then the appropriate pattern is drawn with the CALL PATTERN statements. Lines 630-780 then set the rocket in motion by calling the correct subroutine.

Lines 800-930 fire a shot if the correct firing key is pressed. First the position of the rocket is determined, then its direction. The bullet will then be defined as Sprite #3 or #4 going the same direction as the rocket is pointing, and the speed is dependent on the level of difficulty.

Lines 540-560 check to see if the two rockets collide, if the red rocket is hit by the blue's bullet, and if the blue rocket is hit by the red's bullet. If there is coincidence among any of these pairs of sprites, the program branches to the appropriate subroutines in lines 940-1150. CALL MOTION stops the rockets; then the CALL SPRITE redefines the rocket as a crashed graphics figure, and the rocket falls to the ground. CALL POSITION is used to determine

where the sprite is as it's falling so it can be deleted at the bottom of the screen.

Lines 1190-1230 present the option to try again and branch appropriately.

Rocket Duel

```
90 REM TI EXTENDED BASIC
100 CALL CLEAR
110 DISPLAY AT(4,5):"** ROCKET DUEL **"
120 DISPLAY AT(8,3):"PLAYERS CONTROL ROCKETS
    " :: DISPLAY AT(10,3):"WITH ARROW KEYS."
     :: DISPLAY AT(13,3):"SHOOT WITH 'F' AND
     'H'."
130 CALL MAGNIFY(3):: A1=1 :: A2=5
140 CALL CHAR(96,"00000000F0783F3F3F3F78F000
    0000000000000000F0FFFFF")
150 CALL CHAR(100,"0000000000001037F1F07030
    301010003070E3C78F8F0E0C08")
160 CALL CHAR(104,"0101010103030303030303070
    F0F0C0880808080C0C0C0C0C0C0C0E0F0F0308")
170 CALL CHAR(108,"C0E0703C1E1F0F07030100000
    0000000000000000000080C0FEF8E0C0C0808")
180 CALL CHAR(112,"0000000000000FFFFF0F00000
    0000000000000000F1EFCFCFCFC1E0F")
190 CALL CHAR(116,"0000000000000103070F1F1E3
    C70E0C0008080C0C0E0F8FEC08")
200 CALL CHAR(120,"080C0F0F07030303030303030
    10101011030F0F0E0C0C0C0C0C0C0C08080808")
210 CALL CHAR(124,"0001010303071F3F030100000
    0000000000000000000080C0E0F0F8783C0E0703"
    )
220 CALL CHAR(128,"0000000000000001010000000
    0000000000000000000000808")
230 CALL CHAR(132,"C1E171381C0E07E3E3070E1C3
    871E1C183878E1C3870E0C7C7E070381C8E8783"
    )
240 DISPLAY AT(18,3):"CHOOSE:" :: DISPLAY AT
    (20,6):"1 BEGINNER" :: DISPLAY AT(21,6):
    "2 INTERMEDIATE" :: DISPLAY AT(22,6):"3
    ADVANCED"
250 CALL KEY(0,K,S):: IF (K<49)+(K>51)THEN 2
    50
260 L=2*(K-48)
270 CALL CLEAR :: RANDOMIZE
280 CALL SPRITE(#1,96,7,INT(RND*180),20,0,L,
    #2,112,6,INT(RND*180),240,0,-L)
290 CALL KEY(1,K1,S1):: CALL KEY(2,K2,S2)
300 IF S1=0 THEN 420
```

```
310 IF K1<>5 THEN 330 ELSE IF A1>3 AND A1<8
    THEN A1=A1-1 ELSE IF A1<3 OR A1=8 THEN A
    1=A1+1
320 GOTO 380
330 IF K1<>2 THEN 350 ELSE IF A1>5 THEN A1=A
    1-1 ELSE IF A1<5 AND A1>1 THEN A1=A1+1
340 GOTO 380
350 IF K1+1<>1 THEN 370 ELSE IF A1>3 AND A1<
    7 THEN A1=A1+1 ELSE IF A1<3 OR A1=8 THEN
     A1=A1-1
360 GOTO 380
370 IF K1<>3 THEN 410 ELSE IF A1<5 AND A1>1
    THEN A1=A1-1 ELSE IF A1>5 THEN A1=A1+1
380 IF A1=0 THEN A1=8
390 IF A1=9 THEN A1=1
400 CALL PATTERN(#1,(A1*4)+92):: ON A1 GOSUB
     630,640,650,660,670,680,690,700 :: GOTO
     420
410 IF K1=12 THEN GOSUB 790
420 IF S2=0 THEN 540
430 IF K2<>5 THEN 450 ELSE IF A2>3 AND A2<8
    THEN A2=A2-1 ELSE IF A2<3 OR A2=8 THEN A
    2=A2+1
440 GOTO 500
450 IF K2+1<>1 THEN 470 ELSE IF A2>3 AND A2<
    7 THEN A2=A2+1 ELSE IF A2<3 OR A2=8 THEN
     A2=A2-1
460 GOTO 500
470 IF K2<>2 THEN 490 ELSE IF A2>5 THEN A2=A
    2-1 ELSE IF A2<5 AND A2>1 THEN A2=A2+1
480 GOTO 500
490 IF K2<>3 THEN 530 ELSE IF A2<5 AND A2>1
    THEN A2=A2-1 ELSE IF A2>5 THEN A2=A2+1
500 IF A2=0 THEN A2=8
510 IF A2=9 THEN A2=1
520 CALL PATTERN(#2,(A2*4)+92):: ON A2 GOSUB
     710,720,730,740,750,760,770,780 :: GOTO
     540
530 IF K2=1 THEN GOSUB 910
540 CALL COINC(#1,#2,2.5*L,C):: IF C=-1 THEN
     1090
550 CALL COINC(#1,#4,3*L,C):: IF C=-1 THEN 1
    020
560 CALL COINC(#2,#3,3*L,C):: IF C=-1 THEN 9
    40
570 CALL SOUND(-4000,110,4,-4,4)
580 IF B=0 THEN 600
```

```
590 CALL POSITION(#3,BX,BY):: IF BX<6 OR BX>
    186 OR BY<8 OR BY>250 THEN CALL DELSPRIT
    E(#3):: B=0
600 IF B2=0 THEN 290
610 CALL POSITION(#4,B2X,B2Y):: IF B2X<6 OR
    B2X>186 OR B2Y<8 OR B2Y>250 THEN CALL DE
    LSPRITE(#4):: B2=0
620 GOTO 290
630 CALL MOTION(#1,0,L):: RETURN
640 CALL MOTION(#1,-L*.7,L*.7):: RETURN
650 CALL MOTION(#1,-L,0):: RETURN
660 CALL MOTION(#1,-L*.7,-L*.7):: RETURN
670 CALL MOTION(#1,0,-L):: RETURN
680 CALL MOTION(#1,L*.7,-L*.7):: RETURN
690 CALL MOTION(#1,L,0):: RETURN
700 CALL MOTION(#1,L*.7,L*.7):: RETURN
710 CALL MOTION(#2,0,L):: RETURN
720 CALL MOTION(#2,-L*.7,L*.7):: RETURN
730 CALL MOTION(#2,-L,0):: RETURN
740 CALL MOTION(#2,-L*.7,-L*.7):: RETURN
750 CALL MOTION(#2,0,-L):: RETURN
760 CALL MOTION(#2,L*.7,-L*.7):: RETURN
770 CALL MOTION(#2,L,0):: RETURN
780 CALL MOTION(#2,L*.7,L*.7):: RETURN
790 IF B=1 THEN RETURN
800 CALL POSITION(#1,BA,BB)
810 D=A1 :: N=3 :: R1=BA :: C1=BB :: B=1
820 ON D GOTO 830,840,850,860,870,880,890,90
    0
830 CALL SPRITE(#N,128,2,R1,C1,0,L*2):: RETU
    RN
840 CALL SPRITE(#N,128,2,R1,C1,-L*1.4,L*1.4)
    :: RETURN
850 CALL SPRITE(#N,128,2,R1,C1,-L*2,0):: RET
    URN
860 CALL SPRITE(#N,128,2,R1,C1,-L*1.4,-L*1.4
    ):: RETURN
870 CALL SPRITE(#N,128,2,R1,C1,0,-L*2):: RET
    URN
880 CALL SPRITE(#N,128,2,R1,C1,L*1.4,-L*1.4)
    :: RETURN
890 CALL SPRITE(#N,128,2,R1,C1,L*2,0):: RETU
    RN
900 CALL SPRITE(#N,128,2,R1,C1,L*1.4,L*1.4):
    : RETURN
910 IF B2=1 THEN RETURN
920 CALL POSITION(#2,B2X,B2Y)
930 R1=B2X :: C1=B2Y :: D=A2 :: N=4 :: B2=1
    :: GOTO 820
```

```
940 CALL DELSPRITE(#3,#4)
950 CALL MOTION(#2,0,0):: CALL POSITION(#2,P
    2X,P2Y)
960 CALL SPRITE(#2,132,10,P2X,P2Y,20,0)
970 CALL SOUND(-500,110,4,-5,4)
980 CALL POSITION(#2,P2X,P2Y):: IF P2X<186 T
    HEN 970
990 CALL DELSPRITE(#2,#1)
1000 RED=RED+1
1010 DISPLAY AT(12,4):"RED WON THE DUEL." ::
     GOTO 1160
1020 CALL DELSPRITE(#3,#4)
1030 CALL MOTION(#1,0,0):: CALL POSITION(#1,
     P1X,P1Y)
1040 CALL SPRITE(#1,132,10,P1X,P1Y,20,0)
1050 CALL SOUND(-1000,110,4,-5,4)
1060 CALL DELSPRITE(#1,#2)
1070 BLUE=BLUE+1
1080 DISPLAY AT(12,4):"BLUE WON THE DUEL." :
     : GOTO 1160
1090 CALL MOTION(#1,0,0,#2,0,0):: CALL POSIT
     ION(#1,P1X,P1Y,#2,P2X,P2Y)
1100 CALL SPRITE(#1,132,10,P1X,P1Y,20,0,#2,1
     32,10,P2X,P2Y,20,0)
1110 CALL POSITION(#1,P1X,P1Y):: CALL POSITI
     ON(#2,P2X,P2Y):: IF P1X>186 OR P2X>186
     THEN CALL DELSPRITE(#1,#2)
1120 CALL SOUND(-500,110,4,-6,4)
1130 IF P1X<>0 OR P2X<>0 THEN 1110
1140 CALL DELSPRITE(ALL)
1150 DISPLAY AT(12,4):"BOTH ROCKETS CRASHED.
     " :: GOTO 1160
1160 DISPLAY AT(16,4):"TOTAL SCORE IS:"
1170 DISPLAY AT(18,8):"RED = ";RED
1180 DISPLAY AT(20,8):"BLUE =";BLUE
1190 DISPLAY AT(23,4):"TRY AGAIN? (Y/N)"
1200 A1=1 :: A2=5 :: B=0 :: B2=0 :: C=0
1210 CALL KEY(0,K,S)
1220 IF K=89 THEN 270
1230 IF K<>78 THEN 1210
1240 END
```

9

Mystery Spell

Doug Hapeman

"Mystery Spell" is an educational program which uses sprites to make spelling practice more entertaining. The TI Extended BASIC module is required.

"Mystery Spell" is a teaching aid for parents, teachers, and students. Spelling lessons can be reviewed and then practiced with a simple guessing game. The object of the program is to try to guess the secret word by selecting the correct spelling.

Mystery Spell is similar in concept to the popular Hangman game, but with a different twist. Rather than using the negative symbols used in Hangman, Mystery Spell reinforces the positive emotions with balloons, blackbirds, and a happy face. The smoothness of moving sprites and the addition of music give Mystery Spell an edge of fascination over most versions of Hangman.

Happy Face and Blackbirds

When the game begins, a happy face appears in a little hut surrounded by trees and landscape. The letters of the alphabet appear near the bottom of the screen, and blank spaces representing the secret word appear near the top. Each time a letter is pressed, the happy face moves to the selected letter and indicates if it is a correct choice. For each correct choice, a colored balloon rises to the appropriate place in the secret word and the letter is displayed. If the choice is incorrect, a blackbird descends and lands somewhere on the landscape. Too many blackbirds will cause the player to lose, and the word will be spelled correctly.

The two levels of difficulty are: easy, which permits six incorrect guesses, and difficult, which allows four.

The player may use the 20 words provided within the program or enter a different word list. A user-generated word list may be SAVEd to a storage device.

Screen-Centered Printing

There are several places in the program where variable length words or phrases are centered on the screen. For centering text

with the DISPLAY AT statement, a simple equation can determine the proper column position:

Column = (14 - LEN(L$)/2).

It's like using a typewriter. When you want to center your title, you find the center of the page and count back one-half the length of the title.

So, in TI BASIC you subtract one-half the length of the string variable from one-half the screen width. One-half the screen width using DISPLAY AT is 14; and using CALL HCHAR, 16. The length of the string variable is easily determined by the LENgth function.

Moving Sprites

Moving sprites are a fascinating feature of TI's Extended BASIC. Through an impressive library of subprograms, sprites can easily be called, defined, magnified, and set in motion. They can acknowledge coincidence, change character definition, etc. Because the sprites are controlled by built-in subprograms, even a beginning programmer can use them easily.

Regular characters are located on the screen in a 32-column and 24-row format, making possible a total of 768 screen positions. Sprites, however, are located by dot-row and dot-column positions. Each regular character position is made up of an eight-by-eight grid. Sprites, on the other hand, can be located at any one of the 64 dots in the eight-by-eight character grid. Therefore, there are 192 dot-rows and 256 dot-columns, for a total of 49,152 screen positions.

Mystery Spell uses moving sprites for several special effects. The happy face sprite is first positioned in the little hut. Each time a letter is pressed, it moves to the location of the letter and then back to the hut.

Let's examine just how the happy face movement is achieved. The numeric variables used for determining direction and motion are C for the column, R for the row, V for the vertical motion, and H for the horizontal motion.

The alphabet is displayed on the screen in two neat rows, (A-M) and (N-Z). The ASCII value of the alphabet is 65 for A to 90 for Z. In response to the CALL KEY, any key pressed other than A-Z is ignored. If the letter pressed is less than 78 (the letter N), then the row variable is set for the upper row. Otherwise, the row variable is set for the lower row (line 350). CALL GCHAR deter-

mines whether the letter has previously been chosen (line 360). If not, then the vertical motion is set for downward movement until coincidence is achieved with the row variable, then motion stops (line 390 and subroutine 540).

Knowing which way to move horizontally is determined with another IF-THEN statement (line 400).

Knowing where to stop horizontally presented a more difficult problem. It could have been determined by the process of elimination through a long series of IF-THEN statements. But once again a simple equation came to the rescue (line 410):

C=(K-64)*16+4-208*INT((K-64)/14).

(K-64) gives a number between 1 and 26, depending on which letter has been pressed. It is multiplied by 16, which is two times eight dot-column positions (one for the letter and one for the space). A 4 is added to center the sprite over the appropriate letter. 208*INT((K-64)/14) yields either 0 or 208. ((K-64)/14) yields 0 for (A-M) or 1 for (N-Z). 208 represents 26 character positions (13 letters and 13 spaces in each row) times eight dot positions per character position.

The best way for you to understand how the equation works is to experiment by placing in different values for K. For example, suppose the letter F was pressed.

The ASCII value of F is 70.
C=(70-64)*16+4-208*INT ((70-64)/14)
C=96+4-208*INT6/14
C=96+4-208*0
C=100 The dot-column position for F.

Balloon Motion

The balloon sprite moves from wherever the happy face sprite is located to the appropriate blank in the secret word at the top of the screen. Study the correct guess subroutine (lines 570-600) to see if you can follow the program logic for balloon direction and motion.

Explanation of the Program

Lines	
100-270	REMs, initialization, and introduction.
280-530	Main program loop.
540-560	Subroutine to move happy face.
570-600	Subroutine for correct guess.
610-690	Subroutine for incorrect guess.

700-720	Subroutine for player loses.
730-760	Subroutine for player wins.
770-910	Routine to assign colors and define characters.
920-960	Routine to print screen.
970-1030	Routine for instructions.
1040-1600	Subroutine for music and bird flight.
1610-1820	Routine to create and store your own word list.
2000-2060	Tape or disk error trapping.

Mystery Spell

```
100 REM TI EXTENDED BASIC
110 REM MYSTERY SPELL
120 DIM A$(26),B$(20)
122 ON ERROR 140
125 CALL INIT :: CALL LOAD(-31878,13)
130 REM **INITIALIZATION AND INTRODUCTION**
140 DISPLAY AT(12,5)ERASE ALL:"ONE MOMENT PL
    EASE..." :: GOTO 780
150 DISPLAY AT(7,1)ERASE ALL BEEP:"PRESS
    {3 SPACES}FOR": : :"  1  =  INSTRUCTIONS
    ": :"  2  =  MYSTERY SPELL": :"  3  =  F
    INISH MYSTERY SPELL"
160 DISPLAY AT(23,3):"PLEASE ENGAGE ALPHA LO
    CK" :: CALL KEY(0,K,S):: IF S=0 OR(K<49
    OR K>51)THEN 160 :: ON K-48 GOTO 980,190
    ,170
170 DISPLAY AT(12,5)ERASE ALL BEEP:"THANKS F
    OR PLAYING," :: DISPLAY AT(14,14-LEN(L$)
    /2):L$ :: STOP
190 DISPLAY AT(7,1)ERASE ALL BEEP:"CHOOSE A
    WORD LIST:": : :"  A  =  PRESELECTED WOR
    DS": :"  B  =  CREATE YOUR OWN"
200 CALL KEY(0,K,S):: IF S=0 OR(K<65 OR K>66
    )THEN 200 :: IF K=66 THEN 220
210 PSW=1 :: GOTO 230
220 PSW=0 :: GOTO 1620
230 CALL CLEAR :: RESTORE 940 :: GOTO 930
240 CALL SPRITE(#2,120,2,78,121,0,0):: CALL
    MAGNIFY(3):: CALL SPRITE(#4,136,16,8,128
    ,0,1):: CALL SPRITE(#3,140,2,8,128,0,-2)
250 DISPLAY AT(5,9):"MYSTERY SPELL" :: T=200
     :: GOSUB 1050 :: IF PSW=1 THEN GOTO 184
    0
260 DISPLAY AT(19,1)BEEP:" WHAT IS YOUR NAME
    , PLEASE?" :: DISPLAY AT(23,1):"TYPE NAM
    E, THEN PRESS ENTER"
270 ACCEPT AT(5,9)SIZE(14):L$ :: CALL HCHAR(
    5,7,32,22)
280 REM **MAIN PROGRAM LOOP**
```

```
290 DISPLAY AT(19,1)BEEP:"  CHOOSE THE LEVEL
     OF PLAY"
300 DISPLAY AT(23,1):"{3 SPACES}1) EASY
    {3 SPACES}2) DIFFICULT" :: CALL KEY(0,K,
    S):: IF S=0 OR K>50 OR K<49 THEN 300 ::
    IF K=49 THEN ER=7 ELSE ER=5
310 FOR SP=5 TO 13 :: CALL DELSPRITE(#SP)::
    NEXT SP
320 DISPLAY AT(19,1):" A B C D E F G H I J K
     L M" :: DISPLAY AT(23,1)BEEP:" N O P Q
    R S T U V W X Y Z" :: RANDOMIZE
330 CALL HCHAR(5,3,32,28):: W$=B$(INT(20*RND
    )+1):: F=LEN(W$):: FOR I=1 TO F :: DISPL
    AY AT(5,2*I+14-F):"_" :: NEXT I :: Y=0 :
    : SP=13
340 CALL KEY(0,K,S):: IF S=0 OR(K<65 OR K>90
    )THEN 340 ELSE C=121
350 IF K<78 THEN R=128 ELSE R=160
360 CC=((K-64)*16+16-208*INT((K-64)/14))/8 :
    : CALL GCHAR((R+24)/8,CC,X):: IF X=32 TH
    EN 370 ELSE 390
370 DISPLAY AT(16,14-(8+LEN(L$))/2)SIZE(8+LE
    N(L$))BEEP:" OOPS, ";L$;"," :: DISPLAY A
    T(17,1):" YOU TRIED THAT ONE ALREADY"
380 FOR D=1 TO 500 :: NEXT D :: CALL HCHAR(1
    6,1,33,64):: GOTO 340
390 V=12 :: H=0 :: GOSUB 550
400 IF K<72 OR(K>77 AND K<85)THEN H=-12 ELSE
     H=12
410 V=0 :: C=(K-64)*16+4-208*INT((K-64)/14):
    : GOSUB 550
420 X=0 :: CALL HCHAR((R+24)/8,(C+12)/8,32):
    : FOR I=1 TO F :: IF ASC(SEG$(W$,I,1))<>
    K THEN 450
430 CALL PATTERN(#2,124):: GOSUB 580
440 CALL PATTERN(#2,120):: DISPLAY AT(5,2*I+
    14-F)SIZE(-1):CHR$(K):: X=1 :: Y=Y+1
450 NEXT I :: IF X=1 THEN 470
460 CALL PATTERN(#2,128):: GOSUB 620 :: CALL
     PATTERN(#2,120)
470 H=-H :: C=121 :: GOSUB 550
480 V=-12 :: H=0 :: R=78 :: GOSUB 550
490 IF Y=LEN(W$)THEN GOSUB 740 ELSE 500 :: G
    OTO 510
500 IF ER=1 THEN GOSUB 710 ELSE 340
510 DISPLAY AT(23,1)BEEP:"{5 SPACES}ANOTHER
    WORD? (Y/N)"
520 CALL KEY(0,K,S):: IF S=0 OR K<>89 AND K<
    >78 THEN 520 :: IF K=89 THEN 290
```

```
530 CALL DELSPRITE(ALL):: GOTO 150
540 REM **SUB TO MOVE HAPPY FACE**
550 CALL MOTION(#2,V,H)
560 CALL COINC(#2,R,C,4,Z):: IF Z=0 THEN 560
     ELSE CALL MOTION(#2,0,0):: CALL LOCATE(
    #2,R,C):: RETURN
570 REM **SUB FOR CORRECT GUESS**
580 B=8*(2*I+14-F):: CALL SPRITE(#1,132,14,R
    ,C,(32-R)/8,(B-C)/8)
590 J=2^(1/12):: FOR A=1 TO 25 :: CALL SOUND
    (-40,220*J^A,1):: NEXT A
600 CALL COINC(#1,32,B,6,Z):: IF Z=0 THEN 60
    0 ELSE CALL DELSPRITE(#1):: RETURN
610 REM **SUB FOR INCORRECT GUESS**
620 SP=SP-1 :: ER=ER-1 :: IF ER>4 THEN RR=80
     ELSE RR=50
630 IF ER=6 OR ER=4 THEN C=52
640 IF ER=5 OR ER=1 THEN C=188
650 IF ER=3 THEN C=110
660 IF ER=2 THEN C=132
670 CALL SPRITE(#SP,140,2,1,120,(RR-1)/8,(C-
    120)/8)
680 J=2^(1/12):: FOR A=25 TO 1 STEP -1 :: CA
    LL SOUND(-40,440*J^A,1):: NEXT A
690 CALL COINC(#SP,RR,C,6,Z):: IF Z=0 THEN 6
    90 ELSE CALL MOTION(#SP,0,0):: CALL LOCA
    TE(#SP,RR,C):: CALL PATTERN(#SP,100):: R
    ETURN
700 REM **SUB FOR BLACKBIRDS WIN**
710 CALL HCHAR(19,3,32,28):: DISPLAY AT(19,1
    5-(8+LEN(L$))/2):"SORRY, ";L$;","
720 DISPLAY AT(23,1)BEEP:"THE BLACKBIRDS WIN
     THIS TIME" :: GOSUB 760 :: RETURN
730 REM **SUB FOR PLAYER WINS**
740 CALL HCHAR(19,3,32,28):: DISPLAY AT(19,1
    5-(8+LEN(L$))/2):"GREAT, ";L$;","
750 DISPLAY AT(23,1):"{3 SPACES}THAT'S THE S
    ECRET WORD"
760 CALL HCHAR(5,3,32,28):: FOR I=1 TO F ::
    DISPLAY AT(5,2*I+14-F):SEG$(W$,I,1):: NE
    XT I :: T=180 :: GOSUB 1050 :: RETURN
770 REM **ASSIGN COLORS AND DEFINE CHARACTER
    S**
780 FOR I=0 TO 9 :: CALL COLOR(I,2,8):: NEXT
     I :: CALL COLOR(10,3,8):: CALL COLOR(11
    ,11,8):: CALL COLOR(1,13,8)
800 FOR I=1 TO 25 :: READ C,A$(I):: CALL CHA
    R(C,A$(I)):: NEXT I :: CALL SCREEN(15)::
     GOTO 150
```

```
810 DATA 112,C0C0C0C0C0C0C0C0,113,0303030303
    030303,114,FFFFFFFFFFFFFFFF,115,C0C0C0FF
    FFC0C0C0,116,030303FFFF030303
820 DATA 105,183C3C7E7EFFFFFF,106,FFFFFFFFFF
    FFFFFF,107,FFFFFF7E7E3C3C18,108,071F7FFF
    FF7F1F07,109,C0F0FEFFFFFEF0C0
830 DATA 96,00000000030F3FFF,97,FFFFFFFFFFFF
    FFFF,98,FFFFEFC78383C7EF,99,7F3F1E3C78FC
    FEFF,33,FFFFFFFFFFFFFFFF
840 DATA 91,1F3F7FFFFFFFFFFF,92,F8FCFEFFFFFF
    FFFF,93,00000000C0F0FCFF
850 DATA 120,071820404C88081808884434020180
    7E0180C0232110181011121C2020418E0
860 DATA 124,071820404C888081809F90484423180
    7E0180C023211018101F9091222C418E0
870 DATA 128,071820404C88081808384484020180
    7E0180C023211018101C12112020418E0
880 DATA 132,030F1F3F3F3F3F1F0F0703010102040
    880E0F0F8F8F8F8F0E0C080
890 DATA 136,030F3F7F7F3FFFFFFFFF3F7F7F37070
    100C0CCFEFEFCFFFFFFFFFCFEFEECE080
900 DATA 140,000000000000183D478301000000000
    000000000000018BCE2C1800000000000
910 DATA 100,000101010001030307070703010001O
    1C0E0F0D0C0E0F0F0F8F8F8F0E0C02020
920 REM **PRINT SCREEN**
930 CALL HCHAR(16,1,33,288):: FOR I=1 TO 21
    :: READ R,C,G$ :: DISPLAY AT(R,C)SIZE(-6
    ):G$ :: NEXT I :: GOTO 240
940 DATA 9,12,`[aa\],10,12,qrrrrp,11,12,qrrr
    rp,12,12,qrrrrp,13,14,st,14,14,st,15,14,
    st
950 DATA 9,5,iii,10,4,1jjjm,11,4,1jjjm,12,5,
    kjk,13,6,b,14,6,a,15,6,a
960 DATA 9,22,iii,10,21,1jjjm,11,21,1jjjm,12
    ,22,kjk,13,23,a,14,23,c,15,23,a
970 REM **INSTRUCTIONS**
980 DISPLAY AT(1,8)ERASE ALL:"MYSTERY SPELL"
    : :"{3 SPACES}THE OBJECT OF MYSTERY":"SP
    ELL IS TO GUESS THE SECRET":"WORD."
990 DISPLAY AT(6,4):"WHEN YOU PRESS A LETTER
    ,":"THE HAPPY FACE WILL MOVE TO":"THE SE
    LECTED LETTER AND LET":"YOU KNOW WHETHER
     YOU MADE A"
1000 DISPLAY AT(10,1):"RIGHT OR WRONG CHOICE
     .":"{3 SPACES}A CORRECT CHOICE LAUNCHES
     ":"A BALLOON.  AN INCORRECT ONE":"CAUSE
     S A BLACKBIRD TO LAND."
```

```
1010 DISPLAY AT(14,1):"IF TOO MANY BLACKBIRD
     S LAND,":"YOU WILL LOSE THE GAME.": :"
     {3 SPACES}THERE ARE TWO LEVELS:"
1020 DISPLAY AT(19,1)BEEP:"EASY)  PERMITS 6
     INCORRECT":"{7 SPACES}GUESSES.": :"HARD
     )  PERMITS ONLY 4."
1030 DISPLAY AT(24,6):"**PRESS ANY KEY**" ::
      CALL KEY(0,K,S):: IF S=0 THEN 1030 ELS
     E 190
1040 REM **SUB FOR BLACKBIRD FLIGHT AND THEM
     E MELODY**
1050 R=8 :: FOR SP=5 TO 13 :: C=INT(RND*240)
     +1 :: CALL SPRITE(#SP,140,2,R,C,0,12)::
      R=R+12 :: NEXT SP
1060 CALL SOUND(T,175,0)
1070 CALL SOUND(T,349,0,175,2)
1080 CALL SOUND(T,587,0,175,2)
1090 CALL SOUND(2*T,523,0,440,1,175,2)
1100 CALL SOUND(T,587,0,175,2)
1110 CALL SOUND(2*T,523,0,440,1,185,2)
1120 CALL SOUND(T,196,0)
1130 CALL SOUND(T,330,0,196,2)
1140 CALL SOUND(T,587,0,196,2)
1150 CALL SOUND(2*T,523,0,466,1,196,2)
1160 CALL SOUND(T,587,0,196,2)
1170 CALL SOUND(2*T,523,0,466,1,208,2)
1180 CALL SOUND(T,220,2)
1190 CALL SOUND(T,523,0,440,1,220,2)
1200 CALL SOUND(T,311,2)
1210 CALL SOUND(T,523,0,440,1,311,2)
1220 CALL SOUND(T,294,2)
1230 CALL SOUND(T,494,0,415,1,294,2)
1240 CALL SOUND(T,277,2)
1250 CALL SOUND(T,466,0,392,1,277,2)
1260 CALL SOUND(T,440,0,262,2)
1270 CALL SOUND(T,523,0,262,2)
1280 CALL SOUND(T,587,0,247,2)
1290 CALL SOUND(T,698,0,247,2)
1300 CALL SOUND(2*T,659,0,233,2)
1310 CALL SOUND(2*T,784,0,659,1,131,2)
1320 CALL SOUND(T,880,0,175,2)
1330 CALL SOUND(T,831,0,175,2)
1340 CALL SOUND(T,880,0,262,2,349,2)
1350 CALL SOUND(T,1047,0,262,2,349,2)
1360 CALL SOUND(T,1047,0,220,2)
1370 CALL SOUND(T,880,0,220,2)
1380 CALL SOUND(T,784,0,262,2,349,2)
1390 CALL SOUND(T,698,0,262,2,349,2)
1400 CALL SOUND(T,784,0,233,2)
```

```
1410 CALL SOUND(T,698,0,233,2)
1420 CALL SOUND(T,587,0,294,2,349,2)
1430 CALL SOUND(T,698,0,294,2,349,2)
1440 CALL SOUND(T,698,0,220,2)
1450 CALL SOUND(T,587,0,220,2)
1460 CALL SOUND(T,523,0,262,2,349,2)
1470 CALL SOUND(T,440,0,262,2,349,2)
1480 CALL SOUND(T,392,0,247,2)
1490 CALL SOUND(T,784,0,247,2)
1500 CALL SOUND(T,698,0,294,2,349,2)
1510 CALL SOUND(T,659,0,294,2,349,2)
1520 CALL SOUND(T,587,0,196,2)
1530 CALL SOUND(T,440,0,196,2)
1540 CALL SOUND(T,440,0,233,2,349,2)
1550 CALL SOUND(T,494,0,233,2,349,2)
1560 CALL SOUND(T,523,0,175,2,220,2)
1570 CALL SOUND(T,587,0,175,2,220,2)
1580 CALL SOUND(2*T,659,0,262,2)
1590 CALL SOUND(3*T,698,0,262,2,175,0)
1600 FOR I=1 TO 30 STEP 2 :: CALL SOUND(-T,6
     98,I,262,I,175,I):: NEXT I :: RETURN
1610 REM **CREATE A WORD LIST**
1620 DISPLAY AT(1,4)ERASE ALL:"WORD LIST INS
     TRUCTIONS": : :"  IN THIS SEGMENT YOU M
     AY":"EITHER CREATE A WORD LIST"
1630 DISPLAY AT(6,1):"OR LOAD AN EXISTING ON
     E FROM":"A STORAGE DEVICE.": : :"  WHEN
      CREATING A WORD LIST,":"TYPE EACH WORD
     , THEN PRESS"
1640 DISPLAY AT(12,1):"ENTER.  MAXIMUM WORD
     LENGTH":"IS 13 CHARACTERS.  20 WORDS":"
     MUST BE ENTERED FOR EACH OF":"THE WORD
     LISTS CREATED."
1650 DISPLAY AT(18,3)BEEP:"AS YOU ENTER EACH
      LIST,":"YOU MAY SAVE IT TO A STORAGE":
     "DEVICE FOR FUTURE USE WITH":"MYSTERY S
     PELL."
1660 DISPLAY AT(24,6):"**PRESS ANY KEY**" ::
      CALL KEY(0,K,S):: IF S=0 THEN 1660
1670 DISPLAY AT(7,1)ERASE ALL BEEP:"PRESS
     {3 SPACES}TO": : :"  1  =  CREATE A WOR
     D LIST": :"  2  =  LOAD A WORD LIST": :
     "  3  =  EXIT"
1680 CALL KEY(0,K,S):: IF S=0 OR(K<49 OR K>5
     1)THEN 1680 :: J=0 :: ON K-48 GOTO 1690
     ,1795,190
1690 DISPLAY AT(1,5)ERASE ALL:"ENTER THE WOR
     D LIST:"
1700 I=1 :: C=1 :: FOR A=1 TO 2 :: R=3 :: FO
     R Z=1 TO 10
```

```
1710 ACCEPT AT(R,C)SIZE(-13)BEEP:B$(I):: R=R
     +2 :: I=I+1 :: NEXT Z :: C=15 :: NEXT A
1720 DISPLAY AT(24,1)BEEP:"CORRECT OR CHANGE
      ANY? (Y/N)"
1730 CALL KEY(0,K,S):: IF S=0 OR K<>89 AND K
     <>78 THEN 1730 :: IF K=89 THEN 1700 ::
     J=1 :: GOTO 1795
1740 FOR I=1 TO 20 :: PRINT #1:B$(I):: NEXT
     I :: CLOSE #1 :: GOTO 230
1750 FOR I=1 TO 20 :: INPUT #1:B$(I):: NEXT
     I :: CLOSE #1
1760 DISPLAY AT(11,6)ERASE ALL BEEP:"DO YOU
     WISH TO SEE": :"{4 SPACES}THE WORD LIST
     ? (Y/N)"
1770 CALL KEY(0,K,S):: IF S=0 OR(K<>89 AND K
     <>78)THEN 1770 :: IF K=89 THEN 1780 ELS
     E 230
1780 DISPLAY AT(1,10)ERASE ALL BEEP:"WORD LI
     ST" :: R=3 :: FOR I=1 TO 20 STEP 2 :: D
     ISPLAY AT(R,1):B$(I),B$(I+1):: R=R+2 ::
      NEXT I
1790 DISPLAY AT(24,1):"PRESS ANY KEY WHEN FI
     NISHED" :: CALL KEY(0,K,S):: IF S=0 THE
     N 1790 ELSE 230
1795 ON ERROR 2000
1800 DISPLAY AT(5,6)ERASE ALL BEEP:"WHAT IS
     THE NAME": :"  OF YOUR STORAGE DEVICE?"
     : :"(EXAMPLE: CS1 OR DSK1.WORDS)"
1810 DISPLAY AT(23,1):"PLACE TAPE OR DISK IN
      DEVICE" :: ACCEPT AT(11,3):F$
1815 IF SEG$(F$,1,1)="C" THEN 1842 ELSE OPEN
      #1:F$,INTERNAL,UPDATE,FIXED 50
1820 IF J=0 THEN 1750 ELSE 1740
1830 REM **PRESELECTED WORD LIST**
1840 FOR I=1 TO 20 :: READ B$(I):: NEXT I ::
      GOTO 260
1842 IF J=0 THEN 1846
1844 OPEN #1:F$,OUTPUT,INTERNAL,FIXED 50 ::
     FOR I=1 TO 20 :: PRINT #1:B$(I):: NEXT
     I :: CLOSE #1 :: GOTO 230
1846 OPEN #1:F$,INPUT ,INTERNAL,FIXED 50 ::
     FOR I=1 TO 20 :: INPUT #1:B$(I):: NEXT
     I :: CLOSE #1 :: CALL CLEAR :: GOTO 176
     0
1850 DATA BANANAS,CARROTS,RHUBARB,CABBAGE,TU
     RNIP,BEANS,CORN,CELERY,WATERMELON,ORANG
     ES,APPLES,PEACHES
1860 DATA MUSHROOMS,ONIONS,POTATOES,TOMATOES
     ,GRAPES,PUMPKIN,SQUASH,LEMONS
```

```
2000 PRINT "ERROR...ERROR"
2010 PRINT "CHECK TAPE OR DISK"
2020 PRINT "PROGRAM WILL BEGIN AGAIN"
2030 PRINT "PLEASE WAIT...."
2040 FOR A=1 TO 500
2050 NEXT A
2060 RUN
```

9

Devastator

David R. Arnold
Translated for the TI by Patrick Parrish

"Devastator" is an arcade-style game which exploits the sprite features of the TI. The screen is given a 3-D effect by careful use of the foreground and background colors. TI Extended BASIC required.

"Devastator" is written in Extended BASIC and requires a joystick. Your mission in this fast and exciting game is to prevent the Devastator's craft from destroying Earth.

As the game begins, you find yourself cruising above the ominous Devastator. A guardian ship from the Devastator appears. You must eliminate this alien ship and at least nine others that follow in a given period. If you fail, the Devastator blasts Earth with a lethal laser.

Two levels of difficulty are offered. On either level, you can eliminate the guardian ship by simply positioning your cross hairs over them using the joystick. The main difference between skill levels lies in the size of these guardian ships (which are actually sprites).

The CALL MAGNIFY statement in line 420 produces ships of two sizes. Consequently, on level one, guardian ships are large and can be easily destroyed, while level two features smaller ships which require greater dexterity to eliminate.

The Program

The primary game loop is from lines 450 to 510. The counter W in line 500 is increased each time through the loop. When W reaches 200, the game is over and Earth is either blasted or saved depending on whether you've destroyed the required number of guardian ships.

If the game, as written, is just too easy or too difficult for you on the skill levels offered, vary the time limit (200) to achieve a level of play suited to your ability.

The programming techniques used here will be of aid to you in writing your own programs on the TI. You may notice that program execution appears to pause between the title screen and the appearance of the playfield (background). Actually the play-

field is being set up, but since the foreground and background colors of all characters are defined as black, nothing appears at this point because the screen color is also black. When all characters on the playfield have been printed, color codes are assigned simultaneously using the CALL COLOR statement so that the entire game field appears at once.

Another trick, also achieved with color coding of characters, gives the game a 3-D effect. The Devastator is first printed in lines 220 to 320 using redefined characters from three character sets. By constantly shifting the foreground and background colors of these character sets in line 450, an illusion of movement is produced. Thus, as you watch the screen, you feel as though you were actually circling this colossal ship.

Devastator

```
90 REM TI EXTENDED BASIC
99 REM DEVASTATOR
100 GOTO 150
110 FOR F=12 TO 14 :: CALL COLOR(F,2,1):: NE
    XT F :: RETURN
120 FOR F=10 TO 16 :: CALL SCREEN(F):: NEXT
    F :: CALL SCREEN(2):: RETURN
130 FOR V=1 TO 30 :: CALL SOUND(D1,F1,V,F2,V
    ):: NEXT V :: RETURN
140 FOR ROW=2 TO 7 :: CALL HCHAR(ROW,22,32,7
    ):: NEXT ROW :: RETURN
150 RANDOMIZE
160 DIM E$(13)
170 CALL CLEAR :: CALL SCREEN(2)
180 GOSUB 530
190 GOSUB 1030 :: CALL CLEAR :: CALL SCREEN(
    2)
200 FOR H=2 TO 14 :: CALL COLOR(H,2,2):: NEX
    T H
210 FOR J=1 TO 4 :: FOR I=1 TO 11 :: CALL HC
    HAR(I,INT(RND*28)+3,46):: NEXT I :: NEXT
     J
220 DISPLAY AT(13,1):"``````````a{6 SPACES}b
    ``````````"
230 DISPLAY AT(14,1):"hhhhhhhhhi`{6 SPACES}`
    jhhhhhhhhh"
240 DISPLAY AT(15,1):"ppppppppqh`{6 SPACES}`
    hrpppppppp"
250 DISPLAY AT(16,1):"pppppppqpha``````bhprp
    pppppp"
```
```

```
260 DISPLAY AT(17,1):"``````appihhhhhhhhjppb
    ``````"
270 DISPLAY AT(18,1):"`````a`pqpppppppppppp`
    b`````"
280 DISPLAY AT(19,1):"hhhhi``qpppppppppppppr`
    `jhhhh"
290 DISPLAY AT(20,1):"hhhih`a``````````````b
    `hjhhh"
300 DISPLAY AT(21,1):"hhihha````````````````
    bhhjhh"
310 DISPLAY AT(22,1):"pqhhihhhhhhhhhhhhhhhhh
    hjhhrp"
320 DISPLAY AT(23,1):"qphihhhhhhhhhhhhhhhhhh
    hhjhpr" :: DISPLAY AT(24,1):"ppihhhhhhhh
    hhhhhhhhhhhhhjpp"
330 FOR J=1 TO 2 :: FOR I=12 TO 14 :: CALL H
    CHAR(I,INT(RND*6)+14,46):: NEXT I :: NEX
    T J
340 DISPLAY AT(2,24):CHR$(120)&CHR$(121):: D
    ISPLAY AT(3,23):CHR$(122)&CHR$(136)&CHR$
    (137)&CHR$(123)
350 DISPLAY AT(4,22):CHR$(124)&CHR$(125)&CHR
    $(138)&CHR$(139)&CHR$(125)&CHR$(126)
360 DISPLAY AT(5,22):CHR$(127)&CHR$(125)&CHR
    $(140)&CHR$(141)&CHR$(125)&CHR$(128)
370 DISPLAY AT(6,23):CHR$(129)&CHR$(142)&CHR
    $(143)&CHR$(130)
380 DISPLAY AT(7,24):CHR$(131)&CHR$(132)
390 CALL COLOR(12,6,1):: CALL COLOR(13,6,1):
    : CALL COLOR(14,3,6)
400 FOR F=2 TO 8 :: CALL COLOR(F,16,1):: NEX
    T F
410 CALL SPRITE(#2,108,11,80,80)
420 CALL MAGNIFY(LEVEL):: SPEED=8 :: TOL=30
    :: IF LEVEL=3 THEN TOL=15
430 CALL SPRITE(#1,100,16,100,110)
440 A=9 :: B=10 :: C=11
450 T=A :: A=B :: B=C :: C=T
460 CALL COLOR(A,2,5):: CALL COLOR(B,2,14)::
     CALL COLOR(C,2,7)
470 CALL MOTION(#2,INT(RND*40-20),INT(RND*40
    -20))
480 CALL JOYST(1,X1,Y1):: CALL MOTION(#1,-Y1
    *SPEED,X1*SPEED)
490 CALL COINC(#1,#2,TOL,G):: IF G THEN GOSU
    B 700
500 W=W+1 :: IF W>200 THEN 770
510 GOTO 450
520 REM DEFINE CHARS
```
```

```
530 A$="" :: B$="0102040810204080" :: C$="80
    40201008040201"
540 CALL CHAR(95,B$)
550 FOR I=96 TO 112 STEP 8 :: CALL CHAR(I,A$
    ):: CALL CHAR(I+1,B$)
560 CALL CHAR(I+2,C$):: NEXT I
570 FOR I=0 TO 13 :: READ E$(I):: CALL CHAR(
    120+I,E$(I)):: NEXT I
580 FOR I=0 TO 7 :: READ E$(I):: CALL CHAR(I
    +136,E$(I)):: NEXT I
590 DATA 0000000000000F7F,000000000000F0FE,0
    1030F1F3F7FFFFF
600 DATA 80C0F0F8FCFEFFFF,0001010103030303,F
    FFFFFFFFFFFFFFF
610 DATA 00808080C0C0C0C0,0303030301010100,C
    0C0C0C080808000
620 DATA FF7F3F3F1F0F0703,FFFEFCFCF8F0E0C0,7
    F0F000000000000
630 DATA FEF0000000000000,0800667C18666810
640 DATA E0F07F7F7FFFFFFF,0818F8F8F0F8F0F0,7
    F7F7F3D1C0E0201
650 DATA F0F0908800180000,03070F0F0F070703,F
    0FFFFFEFCFCF8F0
660 DATA 0303010101010101,E0C0C0C080808000
670 CALL CHAR(108,"00073FE2E2E2FFFF667F0C1C0
    000000000E0FC474747FFFF66FE303800000000")
680 CALL CHAR(100,"0000000003040808FF0808040
    3000000808080E0908888FF888890E0808080")
690 RETURN
700 REM ALIEN SHIP DESTROYED
710 CALL DELSPRITE(#2):: CALL MOTION(#1,0,0)
720 CALL SCREEN(15):: CALL SCREEN(10)
730 CALL SCREEN(2):: FOR DVOL=1 TO 24 STEP 4
     :: CALL SOUND(100,-7,DVOL):: NEXT DVOL
740 CALL SCREEN(2)
750 D=D+1 :: CALL SPRITE(#2,108,11,INT(RND*1
    92)+1,INT(RND*256)+1)
760 RETURN
770 IF D<10 THEN 810
780 GOTO 950
790 FOR I=30 TO 1 STEP -2 :: CALL SOUND(-100
    0,-5,I):: NEXT I :: RETURN
800 REM EARTH DESTROYED
810 GOSUB 790 :: FOR I=8 TO 0 STEP -1 :: CAL
    L HCHAR(7+I,25-I,95):: CALL COLOR(8,INT(
    RND*8)+9,1):: NEXT I
820 FOR J=1 TO 40 :: NEXT J
830 FOR I=8 TO 0 STEP -1 :: CALL HCHAR(7+I,2
    5-I,32):: NEXT I
```

```
840 GOSUB 120 :: D1=-100 :: F1=-6 :: F2=110
    :: GOSUB 130 :: GOSUB 120 :: GOSUB 110
    :: GOSUB 140
850 J=0 :: I=0
860 DISPLAY AT(1,23+I):CHR$(133):: DISPLAY A
    T(1,26+J):CHR$(133)
870 DISPLAY AT(2,22+I):CHR$(133)&CHR$(133)::
     DISPLAY AT(2,26+J):CHR$(133)&CHR$(133)
880 DISPLAY AT(3,21+I):CHR$(133)&CHR$(133)&C
    HR$(133):: DISPLAY AT(4,25+J):CHR$(133)&
    CHR$(133)&CHR$(133)
890 DISPLAY AT(5,22+I):CHR$(133)&CHR$(133)::
     DISPLAY AT(5,25+J):CHR$(133)&CHR$(133)&
    CHR$(133):: GOSUB 120
900 DISPLAY AT(6,25+J):CHR$(133):: DISPLAY A
    T(7,23+I):CHR$(133):: DISPLAY AT(7,27+J)
    :CHR$(133)
910 DISPLAY AT(8,22+I):CHR$(133)&CHR$(133)::
     DISPLAY AT(9+J,24):CHR$(133)&CHR$(133)
920 CALL COLOR(13,9,1):: GOSUB 120 :: D1=30
    :: F1=-6 :: F2=110 :: GOSUB 130 :: IF J=
    1 THEN 940
930 I=-1 :: J=1 :: GOSUB 110 :: GOSUB 120 ::
     GOTO 860
940 FOR F=1 TO 100 :: NEXT F
950 CALL DELSPRITE(ALL):: W=0
960 CALL CLEAR :: CALL SCREEN(2):: DISPLAY A
    T(8,1):"ALIEN SHIPS DESTROYED: ";D
970 IF D>HD THEN HD=D
980 DISPLAY AT(13,6):"BEST ROUND: ";HD
990 D=0 :: DISPLAY AT(17,1):"PLAY AGAIN, CAP
    TAIN (Y/N)?"
1000 CALL KEY(0,KEY,ST):: IF ST=0 THEN 1000
1010 IF (KEY=89)+(KEY=121)THEN CALL CLEAR ::
      GOTO 200
1020 DISPLAY AT(21,6):"SO LONG" :: FOR I=1 T
     O 500 :: NEXT I :: STOP
1030 FOR J=2 TO 8 :: CALL COLOR(J,1,1):: NEX
     T J
1040 PRINT "{4 SPACES}D E V A S T A T O R" :
     : PRINT :: PRINT
1050 PRINT "YOUR MISSION IS TO PROTECT" :: P
     RINT "EARTH FROM THE APPROACHING"
1060 PRINT "DEVASTATOR. SHOOT DOWN AT" :: PR
     INT "LEAST 10 GUARDIAN SHIPS TO"
1070 PRINT "ENABLE YOUR COMRADES TO" :: PRIN
     T "DESTROY THE DEVASTATOR."
1080 PRINT :: PRINT "YOU HAVE ONLY LIMITED T
     IME" :: PRINT "IN WHICH TO COMPLETE YOUR"
```

```
1090 PRINT "MISSION. POSITION YOUR" :: PRINT
      "CROSSHAIR WITH THE JOYSTICK."
1100 FOR J=2 TO 8 :: CALL COLOR(J,15,1):: NE
     XT J
1110 PRINT :: PRINT "ENTER YOUR SKILL LEVEL(
     1,2),CAPTAIN?" :: ACCEPT AT(23,10)BEEP
     VALIDATE("12")SIZE(1):LEVEL$
1120 LEVEL=5-VAL(LEVEL$)
1130 GOSUB 790
1140 PRINT :: PRINT :: PRINT "THE DEVASTATOR
      IS APPROACH-"
1150 PRINT "ING. GRAB YOUR JOYSTICK," :: PRI
     NT "AND PREPARE TO DO BATTLE."
1160 FOR I=1 TO 750 :: NEXT I
1170 RETURN
```

9

Mosaic Puzzle

Rick Rothstein

Before Rubik's Cube became so popular, sliding square puzzles used to challenge the mind. "Mosaic Puzzle" is a computer version of the square puzzle. This computer game goes one better than the original by allowing head-to-head competition through an option that restores the original game board. Requires TI Extended BASIC.

The object of "Mosaic Puzzle" is to slide the blocks about, one at a time, to bring them to one of several preselected patterns. Some patterns that you can try to duplicate are given in Table 1.

When the game is run, you have the option of requesting either letters (A-O) or numbers (1-15) within a 4-by-4 frame. Once you have entered your choice, the game board with its lettered or numbered blocks appears in a scrambled order.

Move the lettered or numbered blocks around the game board with a joystick or the keyboard (E,S,D, and X keys). You actually have a choice of moving either the free space (hole) or the labeled blocks. The game is initially set to move the free space, but by pressing I (note the appearance of the left-right arrow symbol in the lower-left corner of the screen), you can move the labeled blocks.

Each move that you make is tallied and the total number of moves is given at the bottom of the screen. Moves are normally accompanied by a sliding noise and the note symbol will appear at the lower-right corner of the screen. If this noise becomes annoying, press N and the noise will cease and the note will disappear.

At certain times during the game, you may wish to retrace your previous moves. Press − (minus sign) or hit the fire button to step back through each preceding move. With this option, a maximum of 250 moves can be recalled.

Once you've achieved the desired preselected pattern from its scrambled beginnings, you can restore the original game board pattern by pressing FCTN (REDO).

Other options available to you are listed in Table 2.

Well, there you have it. A simple yet challenging game that will keep all puzzle fans intrigued for hours. And if there is more than one puzzle fan in the family, they can compete to see who can copy the pattern in the least number of moves.

Table 1. Possible Patterns For Mosaic Puzzle

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 HORIZONTAL	1 5 9 13 2 6 10 14 3 7 11 15 4 8 12 VERTICAL	7 8 9 10 6 1 2 11 5 4 3 12 15 14 13 SPIRAL
1 2 3 4 12 13 14 5 11 15 6 10 9 8 7 PERIPHERAL	12 2 1 15 7 9 10 4 11 5 6 8 14 13 3 ADDS TO THIRTY	A B C D E F G H I J K L M N O HORIZONTAL
A E I M B F J N C G K O D H L VERTICAL	G H I J F A B K E D C L O N M SPIRAL	A B C D L M N E K O F J I H G PERIPHERAL

Table 2. Game Options

Keystroke	Description
FCTN [BACK]	Return to the letter or number option menu.
FCTN [REDO]	Restore original game board.
FCTN [BEGIN]	Starts a new game.
FCTN [ERASE]	Ends the program.
– or fire button	Step back one move.
N	Turn sound on or off.
I	Returns to moving space or block.

Mosaic Puzzle

```
100 REM TI EXTENDED BASIC
190 CALL MAGNIFY(4):: RANDOMIZE :: DIM TILE(
    16),TEMP(16):: TEMP(16)=16 :: FR=153
200 CALL CLEAR :: CALL SCREEN(11):: CALL CHA
    RSET :: CALL CHAR(35,"0",71,RPT$("0",12)
    &"FF0000FF")
```

```
210 DISPLAY AT(1,9):RPT$("G",12):: DISPLAY A
    T(2,9):"OPTION##MENU"
220 DISPLAY AT(3,9):RPT$("H",12):: DISPLAY A
    T(10,4):"PRESS####FOR" :: DISPLAY AT(11,
    4):"HHHHH####HHH"
230 DISPLAY AT(13,6):"1######NUMBER#PUZZLE"
    :: DISPLAY AT(16,6)BEEP:"2######LETTER#P
    UZZLE"
240 WASTE=RND :: CALL KEY(0,K,ST):: IF ST=0
    THEN 240
250 IF K=49 THEN I=0 ELSE IF K=50 THEN I=1 E
    LSE IF K=7 THEN 700 ELSE CALL SOUND(
150,110,0):: GOTO 240
260 CALL CLEAR :: CALL SCREEN(4):: IF I=0 TH
    EN RESTORE 710 ELSE RESTORE 790
270 FOR I=80 TO 136 STEP 4 :: READ A$,B$ ::
    CALL CHAR(I,"FFFF"&A$&"FFFFFFFF"&B$&"FFF
    F"):: NEXT I :: CALL CHAR(140,RPT$("0",6
    4))
280 CALL CHAR(71,"000000000000000003030303030
    3030300000FFFF")
290 CALL CHAR(74,"00003F3F303030300000FCFC0C
    0C0C0C303030303F3F00000C0C0C0CFCFC",78,R
    PT$("0C",8)&"00000000FFFF")
300 RESTORE 870 :: FOR I=38 TO 47 :: READ A$
     :: CALL CHAR(I,A$):: NEXT I
310 CALL COLOR(5,15,2,6,15,2):: CALL HCHAR(2
    ,7,71,20):: DISPLAY AT(3,5):"GJ"&RPT$("I
    ",16)&"KG"
320 FOR I=4 TO 16 STEP 4 :: DISPLAY AT(I,1):
    RPT$("####GH"&RPT$("G",16)&"NG####",4)::
     NEXT I
330 DISPLAY AT(20,5):"GL"&RPT$("O",16)&"MG"
    :: CALL HCHAR(21,7,71,20)
340 GOSUB 890 :: FOR I=1 TO 15 :: TILE(I)=I
    :: NEXT I
350 FOR J=1 TO 15 :: R=1+INT(RND*(16-J)):: T
    EMP(J)=TILE(R):: TILE(R)=TILE(16-J):: NE
    XT J
360 N=0 :: FOR I=1 TO 14 :: FOR J=1 TO 15-I
    :: IF TEMP(I)>TEMP(I+J)THEN N=N+1
370 NEXT J :: NEXT I :: IF N/2<>INT(N/2)THEN
     TEMP(16)=TEMP(15):: TEMP(15)=TEMP(14)::
     TEMP(14)=TEMP(16):: TEMP(16)=16
380 FOR I=1 TO 16 :: TILE(I)=TEMP(I):: NEXT
    I :: N=0 :: SP=16
390 FOR I=22 TO 124 STEP 34 :: FOR J=62 TO 1
    64 STEP 34 :: N=N+1 :: CALL LOCATE(#TILE
    (N),I,J):: NEXT J :: NEXT I
```

```
400 MOVE$="" :: TOTAL=0 :: DISPLAY AT(24,1):
    CHR$(32+6*DIR)&"#####()*+#,-(.+/:#0#####
    ##"&CHR$(39-7*NO):: CALL SOUND(150,666,0
    ):: GOTO 430
410 CALL SOUND(150,110,0)
420 WASTE=RND :: IF K=73 OR K=78 OR K=105 OR
     K=110 THEN 450
430 CALL KEY(1,KK,ST):: IF KK=18 THEN 670 EL
    SE CALL JOYST(1,X,Y):: IF ABS(X)+ABS(Y)=
    8 OR X+Y=0 THEN 450
440 IF X/4=(-1)^(1-DIR)THEN 510 ELSE IF X/4=
    (-1)^(2-DIR)THEN 550 ELSE IF Y/4=(-1)^(2
    -DIR)THEN 630 ELSE IF Y/4=(-1)^(1-DIR)TH
    EN 590
450 CALL KEY(0,K,ST):: IF ST=0 THEN 430 ELSE
     IF K=45 THEN 670 ELSE IF K=83-HORZ OR K
    =115-HORZ THEN 510
460 IF K=68+HORZ OR K=100+HORZ THEN 550 ELSE
     IF K=88-VERT OR K=120-VERT THEN 590 ELS
    E IF K=69+VERT OR K=101+VERT THEN 630
470 IF K=6 THEN CALL HCHAR(24,4,32,26):: GOS
    UB 890 :: GOTO 380 ELSE IF K=7 THEN 700
480 IF K=15 THEN CALL DELSPRITE(ALL):: GOTO
    200 ELSE IF K=14 THEN CALL HCHAR(24,
    4,32,26):: GOTO 340 ELSE IF ST=-1 THEN 420
490 IF K=78 OR K=110 THEN NO=1-NO :: FR=153+
    NO*30000 :: CALL HCHAR(24,30,39-7*NO)::
    GOTO 420
500 IF K=73 OR K=105 THEN HORZ=15-HORZ :: VE
    RT=19-VERT :: DIR=1-DIR :: CALL HCHAR(24
    ,3,32+6*DIR):: GOTO 420 ELSE 410
510 IF SP=1 OR SP=5 OR SP=9 OR SP=13 THEN 41
    0 ELSE SP=SP-1 :: CALL POSITION(#TILE(SP
    ),ROW,COL):: CALL SOUND(4000,FR,14*NO)
520 IF MINUS=0 THEN MOVE$="L"&MOVE$ ELSE MIN
    US=0
530 FOR I=COL TO COL+34 STEP 2 :: CALL LOCAT
    E(#TILE(SP),ROW,I):: NEXT I
540 TILE(SP+1)=TILE(SP):: TILE(SP)=16 :: CAL
    L SOUND(-1,FR,30):: GOTO 690
550 IF SP=4 OR SP=8 OR SP=12 OR SP=16 THEN 4
    10 ELSE SP=SP+1 :: CALL POSITION(#TILE(S
    P),ROW,COL):: CALL SOUND(4000,FR,14*NO)
560 IF MINUS=0 THEN MOVE$="R"&MOVE$ ELSE MIN
    US=0
570 FOR I=COL TO COL-34 STEP -2 :: CALL LOCA
    TE(#TILE(SP),ROW,I):: NEXT I
580 TILE(SP-1)=TILE(SP):: TILE(SP)=16 :: CAL
    L SOUND(-1,FR,30):: GOTO 690
```

```
590 IF SP>12 THEN 410 ELSE SP=SP+4 :: CALL P
    OSITION(#TILE(SP),ROW,COL):: CALL SOUND(
    4000,FR,14*NO)
600 IF MINUS=0 THEN MOVE$="D"&MOVE$ ELSE MIN
    US=0
610 FOR I=ROW TO ROW-34 STEP -2 :: CALL LOCA
    TE(#TILE(SP),I,COL):: NEXT I
620 TILE(SP-4)=TILE(SP):: TILE(SP)=16 :: CAL
    L SOUND(-1,FR,30):: GOTO 690
630 IF SP<5 THEN 410 ELSE SP=SP-4 :: CALL PO
    SITION(#TILE(SP),ROW,COL):: CALL SOUND(4
    000,FR,14*NO)
640 IF MINUS=0 THEN MOVE$="U"&MOVE$ ELSE MIN
    US=0
650 FOR I=ROW TO ROW+34 STEP 2 :: CALL LOCAT
    E(#TILE(SP),I,COL):: NEXT I
660 TILE(SP+4)=TILE(SP):: TILE(SP)=16 :: CAL
    L SOUND(-1,FR,30):: GOTO 690
670 IF MOVE$="" THEN 410 ELSE MINUS=1 :: K=A
    SC(MOVE$):: MOVE$=SEG$(MOVE$,2,250):: TO
    TAL=TOTAL-2
680 IF K=82 THEN 510 ELSE IF K=76 THEN 550 E
    LSE IF K=85 THEN 590 ELSE IF K=68 THEN 6
    30
690 TOTAL=TOTAL+1 :: DISPLAY AT(24,19)SIZE(4
    )BEEP:TOTAL :: MOVE$=SEG$(MOVE$,1,250)::
     GOTO 420
700 CALL DELSPRITE(ALL):: CALL CLEAR :: END
710 DATA FEFEFEFEFEFEFEFEFEFEFEFE,7F7F7F7F7F
    7F7F7F7F7F7F7F,F8F0F3FFFFFFFEFCF8F1F0F0,
    1F0FCFCF8F1F3F7FFFFF0F0F
720 DATA F8F0F3FFFFFEFEFFFFF3F0F8,1F0FCFCF8F
    1F1F8FCFCF0F1F,FFFFFEFCF8F1F0F0FFFFFFFF,
    8F0F0F4FCFCF0F0FCFCFCFCF
730 DATA F0F0F3F3F0F0FFFFFFF3F0F8,1F1FFFFF3F
    1F8FCFCF8F1F3F,FCF8F1F3F0F0F1F3F3F1F8FC,
    1F1FFFFF3F1F8FCFCF8F1F3F
740 DATA F0F0F3FFFFFFFFFEFEFEFEFE,0F0FCF8F9F
    1F3F3F7F7F7F7F,FCF8F9F9FCF8F1F3F3F1F8FC,
    3F1F9F9F3F1F8FCFCF8F1F3F
750 DATA FCF8F1F3F3F1F8FCFFFFF8F8,3F1F8FCFCF
    8F0F0FCF8F1F3F,CFCECCCCCCCCCCCCCCCCECF,
    0F0763F3F3F3F3F3F363070F
760 DATA F3F3F3F3F3F3F3F3F3F3F3F3,CFCFCFCFCF
    CFCFCFCFCFCFCF,CECCCCCFCFCFCFCFCECCCCCC,
    0703F3F3E3C78F1F3F7F0303
770 DATA CECCCCCFCFCFCFCFCFCCCCCE,0703F3F3E3
    8787E3F3F30307,CFCFCFCFCECCCCCCCFCFCFCF,
    E3C3831333730303F3F3F3F3
```

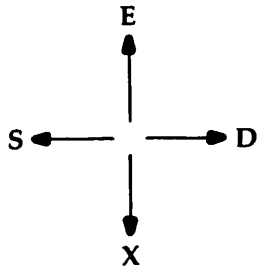
```
780 DATA CCCCCCCCCCCCCFCFCFCCCCCE,0707FFFF0F
    07E3F3F3E3070F
790 DATA FEFCF8F1E3E7E0E0E7E7E7E7,7F3F1F8FC7
    E70707E7E7E7E7,E0E0E7E7E7E0E0E7E7E7E0E0,
    1F0FC7E7C70F0FC7E7C70F1F
800 DATA F8F0E3E7E7E7E7E7E7E3F0F8,0F07E7FFFF
    FFFFFFFFE7070F,E0E0E7E7E7E7E7E7E7E7E0E0,
    3F1F8FC7E7E7E7E7C78F1F3F
810 DATA E0E0E7E7E7E0E0E7E7E7E0E0,0707FFFFFF
    1F1FFFFFFF0707,E0E0E7E7E7E0E0E7E7E7E7E7,
    0707FFFFFF1F1FFFFFFFFFF
820 DATA F8F0E3E7E7E7E7E7E7E3F0F8,0F07E7FFFF
    FF8787E7E70707,E7E7E7E7E7E0E0E7E7E7E7E7,
    E7E7E7E7E70707E7E7E7E7E7
830 DATA F8F8FEFEFEFEFEFEFEFEF8F8,1F1F7F7F7F
    7F7F7F7F7F1F1F,FFFFFFFFFFFFFFFFE7E3F0F8,
    E7E7E7E7E7E7E7E7E7C70F1F
840 DATA E7E7E7E6E4E0E0E0E6E7E7E7,C78F1F3F7F
    FFFF7F3F1F8FC7,E7E7E7E7E7E7E7E7E7E7E0E0,
    FFFFFFFFFFFFFFFFFFFF0707
850 DATA E7E3E1E0E4E6E7E7E7E7E7E7,E7C7870727
    67E7E7E7E7E7E7,E7E3E1E0E4E6E6E7E7E7E7E7,
    E7E7E7E7E76767270787C7E7
860 DATA F8F0E3E7E7E7E7E7E7E3F0F8,1F0FC7E7E7
    E7E7E7E7C70F1F
870 DATA 3060FF0000FF060C,0E09080868F8F86,00
    446C54444444,007C444444447C,004444444428
    1,007C407840407C
880 DATA 00446454544C44,00444444444438,00784
    478444478,00784444784844
890 FOR I=1 TO 16 STEP 2 :: CALL SPRITE(#I,7
    6+4*I,16,193,1,#I+1,80+4*I,11,193,1):: N
    EXT I :: RETURN
```

A

Appendix A: Arrow Keys

Appendix A: Arrow Keys

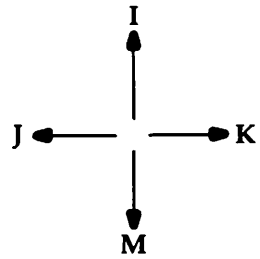
Many of the games in this book use the arrow keys to move an object around. The standard arrow keys on the TI-99/4A are marked on the fronts of the keys. They are:



When you are typing in a program or editing, you can use the arrow keys for various editing features by pressing the FCTN (function) key plus the desired arrow key. In games, however, press the arrow keys without the FCTN key (and on the TI-99/4 press the keys without the SHIFT key).

Some games may allow the diagonal directions.

In a two-player game, one player plays on the regular arrow keys, and the other player uses these keys:



If you have the TI-99/4 console (rectangular keys), an overlay is available that designates the arrow keys for both players. An overlay is *not* available for the TI-99/4A console.

B

Appendix B: Typing in Games

Appendix B: Typing in the Games

The games in this book should be copied exactly as they are listed with one exception. You may encounter braces enclosing a specified number of spaces, i.e.:

```
PRINT "GAME {4 SPACES} OVER"
```

In these (and only these) instances, type the appropriate number of spaces, but do ***not*** type the braces or the words. SAVE a copy, then RUN—and have fun!

If a game is in Extended BASIC, the first statement in the listing will be

```
REM TI EXTENDED BASIC
```

This means you *must* have the TI Extended BASIC command module to use the game.

All the games in this book will work with the disk system. Before you start typing in a game or before you load a previously saved program, use the following commands to clear extra memory:

```
CALL FILES(1)    (ENTER)
NEW              (ENTER)
(now proceed as usual)
```

You may use the NUM command to have the computer automatically number the lines as you type. The last statement of most games is END.

Watch carefully for the difference between zero and the letter O. The letter O is used in words such as ON, GOTO, GOSUB, FOR, TO, and messages within quotes. The number zero is used in series of numbers, in graphics character definition strings (found in DATA statements and in CALL CHAR statements), and in CALL KEY (0,K,S).

Keep the ALPHA LOCK key down at all times except when you're typing lowercase letters and some symbols in PRINT statements or when you're using a joystick. Release the ALPHA LOCK key for the lowercase letters which appear as small capital

letters. Type the symbols on the fronts of the keys by pressing FCTN and the appropriate key.

Take special care in typing the DATA statements. Make sure you have the right number of commas, and make sure the commas are between the numbers correctly as shown in the listings.

The double quote marks "" are pressed twice with no space between them. The ,,, in DATA statements indicates null strings to be read; be sure to count the commas.

Be careful when you're typing double letters. Also, be sure not to hold a key down, or you will get extra, repeated characters.

About every ten lines it's a good idea to compare the listing with what you have typed on the screen. Make sure you're on the correct line—sometimes sections of code are similar, and it's easy to accidentally skip a few lines.

SAVE your game every so often as you are typing, at least every 20 minutes or so. If would be disastrous to work a long time on typing and then have the power suddenly fail. If you use cassettes, use two cassettes. The power could fail while you are in the process of a SAVE, and it's better not to be saving over your only other copy. If you use diskettes, always keep the diskette out of the disk drive except during actual saving or loading. A lightning strike could ruin the whole diskette if it is in the drive.

C

Appendix C: Debugging

Appendix C: Debugging

Every effort has been made to make sure these programs are in good working order. Persons other than the original author have tested them for clarity and "user-proofing." The listings were taken directly from the computer to avoid typing errors. Although we hope the listings are error free, the possibility of errors exists.

If you do find you are having difficulty getting a program to run correctly—*please*, before you call us names or kick the computer, try some of the ideas listed below to help you solve programming problems.

1. Check DATA statements. Most typing errors seem to be in DATA statements. Often a DATA statement contains a series of numbers. Type the numbers carefully, making sure all commas are exactly as shown in the listing. Do *not* end a DATA statement with a comma. If there is a series of commas with nothing between them, count accurately. DATA 6,,7 is not an error; nothing between commas indicates a null. The double quotes "" do *not* have a space between them.

2. Check line numbers in GOTO, ON . . . GOTO, ON . . . GOSUB, GOSUB, and IF . . . THEN statements. In the ON . . . GOTO and ON . . . GOSUB statements, the line numbers are those that the author intended. Subroutines may be entered at several points.

3. Spaces between line numbers is ON . . . GOTO and ON . . . GOSUB statements may cause errors. Do *not* type spaces between the line numbers. If you have accidentally typed a space, it does not always work to DELete it. If you LIST the line, the spaces may not be there but could still cause problems. The solution is to retype the line.

4. Check for zeros. Character definition strings require a zero, not the letter O. Examples are:

```
150 CALL CHAR(96, "FF00FF00")
180 DATA 0103070F,F0F0F
```

In CALL KEY(0,K,S) statements, the first character within the

parentheses is zero. You could use the variable letter O, but to avoid confusion in listings, the letter O has not been used as a variable name in this book.

5. If the graphics seem to be placed wrong on the screen, check CALL HCHAR and CALL VCHAR statements. If high-resolution characters are misshapen, check the CALL CHAR statements or related DATA statements. Check variables leading up to CALL HCHAR and CALL VCHAR statements.

6. RUN the program. If it stops with an error message, LIST the line in the message. Check that line for syntax—spelling, parentheses, and commas. Compare your error message with the list of error messages in the *User's Reference Manual* to get hints of the problem. If the line seems okay, check any variables it may use. You may PRINT the variables to get the current values when the program stopped.

For example, if you get BAD VALUE IN 300, and line 300 is CALL HCHAR(R,C,42), try PRINT R and PRINT C.

R must be a row number from 1 to 24, and C must be a column number from 1 to 32. If the variable exceeds acceptable limits, LIST the lines leading up to line 300 to try to trace how you got there with bad numbers.

At times nothing will be wrong with the particular line that is listed in the error message, but previously executed lines will be the ones causing problems. Be sure to check line numbers.

7. PRINT values of variables at strategic places in the program by adding a few PRINT statements. This will scroll other printing and graphics, but you can see how the variables change in value.

8. Try the command TRACE in particular coding positions. TRACE will print the line numbers being executed as the program is run. This will cause scrolling, and the graphics and other printing will be messed up, but you can follow the program logic. Use UNTRACE to get back to normal.

Index

If you've enjoyed the articles in this book, you'll find the same style and quality in every monthly issue of **COMPUTE!** Magazine. Use this form to order your subscription to **COMPUTE!.**

For Fastest Service,
Call Our **Toll-Free** US Order Line

800-334-0868

In NC call 919-275-9809

COMPUTE!
P.O. Box 5406
Greensboro, NC 27403

My Computer Is:
☐ PET ☐ Apple ☐ Atari ☐ VIC ☐ Other _____ ☐ Don't yet have one...

☐ $20.00 One Year US Subscription
☐ $36.00 Two Year US Subscription
☐ $54.00 Three Year US Subscription

Subscription rates outside the US:
☐ $25.00 Canada
☐ $38.00 Europe, Australia, New Zealand/Air Delivery
☐ $48.00 Middle East, North Africa, Central America/Air Mail
☐ $68.00 Elsewhere/Air Mail
☐ $25.00 International Surface Mail (lengthy, unreliable delivery)

Name _____

Address _____

City _____ State _____ Zip _____

Country _____

Payment must be in US Funds drawn on a US Bank; International Money Order, or charge card.

☐ Payment Enclosed ☐ VISA
☐ MasterCard ☐ American Express
Acct. No. _____ Expires ____/____

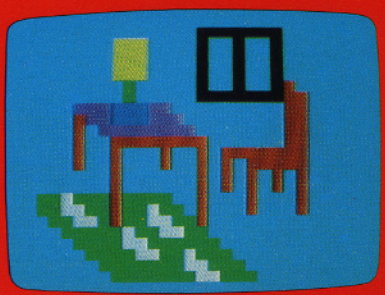
17-5

COMPUTE!'s
First Book of TI Games

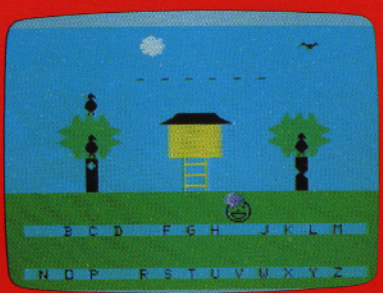
C. Regena, the author of *COMPUTE!'s Programmer's Reference Guide to the TI-99/4A* and the TI columnist for *COMPUTE!* Magazine, has assembled 14 never-before-published games plus 16 of the best games from *COMPUTE!*, complete and ready to type in on your TI-99/4A or TI-99/4 home computer.

Since you can follow the complete BASIC listing, you can see for yourself how programmers create games. Also, several chapters are devoted to helping you learn how to program your own games.

Superchase



Joystick Drawing



Mystery Spell

Mosaic Puzzle

ISBN 0-942386-17-5

$12.95